(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 September 2002 (06.09.2002)

PCT

(10) International Publication Number
WO 02/069541 A2

(51) International Patent Classification⁷: H04L

(21) International Application Number: PCT/US02/01582

(22) International Filing Date: 17 January 2002 (17.01.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/262,049 17 January 2001 (17.01.2001) US
60/262,519 17 January 2001 (17.01.2001) US

(71) Applicant *(for all designated States except US)*: **DMIND** [US/US]; 161 Maiden Lane, New York, NY 10038 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **KWON, Thomas, C.** [—/US]; New York, NY (US). **CAGANELLO, Ronald, B.** [—/US]; Westchester, NY (US). **EKSTROM, Masimillian** [—/US]; Brooklyn, New York (US).

(74) Agents: ULRICH, Lisa, J. et al.; Darby & Darby P.C., 805 Third Avenue, New York, NY 10022-7513 (US).

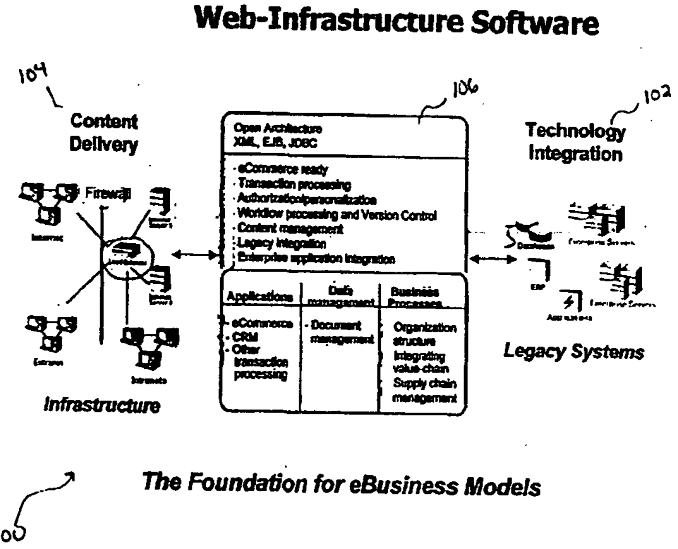
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR GENERATION AND MANAGEMENT OF CONTENT AND SERVICES ON A NETWORK



(57) **Abstract:** The present invention is directed to a rapid creation, provisioning, and management of application services and content services over a distributed computer network. The present invention is a fully componentized, XML-native application for content management and application integration over a distributed computer network. The present invention utilizes a single browser-based interface to: modify Web pages using Extensible Style Sheets (XSL); enable Java applications and external modules; modify user permissions; edit, delete, move, and add pages to Web site; and manage users, user roles and system behaviors.

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

METHOD AND SYSTEM FOR GENERATION AND MANAGEMENT OF CONTENT AND SERVICES ON A NETWORK

CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of priority from provisional U.S. Patent Application 60/262,519 filed on January 17, 2001, and U.S. Application 60/262,049 filed on January 17, 2001, the disclosure of both are incorporated herein by reference in its entirety

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates generally to software for enabling e-Business. More particularly, the present invention relates to software that provides an extensible platform for content management, Enterprise Application Integration ("EAI"), and Enterprise Information Portals ("EIP") for businesses operating over the Internet.

Description of Related Art

In the early days of the Web, sites were primarily static content repositories with little or no transactional components. Thus content consisted of HTML files, images, and downloadable documents like presentations or PDF files. Content managers were typically "webmasters" who had knowledge of HTML, scripting languages, image processing, and Web server software and hardware. Companies often outsourced the content management of their Web sites given the breadth of expertise required, the static nature of the content, and the fact that it was not critical to business operations.

As the technology and importance of the Web have evolved, what is considered "content" has changed dramatically. Web sites are now the central repository for much richer and business-critical content including business processes and rules, software components, data sources, transactional components, visitor profiles, product catalogs, and other data. Such varied content types require very different skill sets among the content managers, many of whom have specialized knowledge that cannot be transferred outside of the organization. Hence it has become infeasible to outsource content management in most modern, business-critical Web sites.

As comparison of prior art solutions with the present invention is depicted in Table A. Table A compares the features within the present invention with two prior art devices, Interwoven's Teamsite and Vignette's V/5.

Functionality	Present Invention	Interwoven Teamsite	Vignette VI5
XML/XSL	Pure XML with XSL transforms	Support for XML, no XSL	Support for XML, no XSL
Content Management	Fully configurable, multi-level approval, version control, extensible using XML and 3rd party apps	Template engine added on top of workflow/versioning development system	Fully configurable, multi-level approval, versioning, extensible using 3rd party apps
Application Integration	Information portal-like functionality, wrap any external data in XML for presentation and management	Limited with new portal Add-on	None that can be managed through templating engine
Personalization	Fully customizable, down to individual page elements - rules, profile and behavior based	None, requires integration with 3rd party application	Customizable - rules, profile and behavior based
Architecture	Fully component based, J2EE, XML, using industry standard Java application servers and databases	Proprietary file system for site storage, database storage of some assets	Vignette Application Foundation (VAF) supports ASP and JSP development. Full package comprises 6 separate applications.
Extensibility	Full set of published API's and SDK available for Java. Integrates with any open API, EJ13, Messaging or XML bridge	Limited to reconfiguring existing functionality only, using command-line versions of function in custom scripts; no APIs	API set for C++ and proprietary scripting language for limited content management functions
Toolset	Java code input and compiler (SDK), message handler, XML schema, directory pool and XSL handlers, plus	None for existing functionality, interface exists for staging and deploying content, source level differencing	Many interfaces to access existing functionality

Functionality	Present Invention	Interwoven Teamsite	Vignette VI5
	interfaces for existing functionality		
Strengths	Component-based, scalable, extensible, robust toolset, cost, use of XML and XSL	Collaborative Web development, versioning	Strong in publishing industry. End-to-end solution with all applications combined

TABLE A

It would be ideal if users managed the Web site content through the Web site (intranet, extranet and/or internet). The Web site would act as the portal or console through which content and application services are deployed. By managing content through the Web site, the creation, deployment and management of disparate content types would be consolidated in one unified Web-based interface. In addition, it would be ideal if the management platform associated with the unified Web-based interface offered an easy extension of internal services while at the same time simplifying integration with external services.

The present invention responded to this need to manage, create and provision complex transactional Web-based services by developing a unified content and transaction management platform. No prior art solution utilizes Extensible Markup Language ("XML") along with Extensible Stylesheet Language ("XSL") and the architectural purity of Java and "EJB" to create a unified Web-based content and application management platform.

SUMMARY OF THE INVENTION

Some objectives of the present invention include, but are not limited to: providing a unified Web interface to access, enable, manage and personalize enterprise data, legacy systems, applications and content; creating a seamless and standardized Web gateway for delivering varied content to any channel; empowering organizations to Web-enable information that was previously inaccessible; modernizing existing business methods and improving existing business methods that can benefit from efficiencies introduced by Web technologies; reducing Web operational and labor costs; and expanding revenue and market opportunities.

To best utilize the Web for business purposes, businesses need to integrate and centralize their existing information assets into transactional Web-enabled portals that can serve the varying needs of internal staff, partners and clients. The flexibility and scalability of the present invention meets this need with an open architecture module deployment platform and a rich set of Application Programming Interfaces ("APIs"). The present invention's module deployment platform allows third-party systems to extend basic application behavior in at least three ways: outsourcing content creation and maintenance, outsourcing lookup and retrieval of pages and outsourcing the handling of application events. Prior art systems can not harness and orchestrate distributed information management as easily and as seamlessly as the present invention. Moreover, the present invention reduces Internet operational costs by sixty percent and enables a ninety percent reduction in labor and time by provisioning Internet services. For example, the present invention can enable a conglomerate servicing over 1,200 contributors to manage its global Intranets and Extranets with only 2 employees.

The modules of the present invention are built on top of a rich set of APIs that the application exposes through Enterprise Java Beans ("EJBs"), the industry-standard for distributed transactional components. These APIs are backed by comprehensive security and highly scalable performance. Leveraging these APIs, each module can focus on the specifics of its business logic and on meeting application goals. With its painless deployment process, robust and convenient APIs and strict adherence to leading industry standards, the present invention offers an open-ended rapid development platform unmatched in any prior art product.

The present invention uses industry standard technologies rather than closed, proprietary conventions. These industry standard technologies include, but are not limited to J2EE, XML, XSL, EJB, JNDI, JMS, and JDBC. By implementing industry standard technologies, functionality can be easily extended, flexibility in selecting software and hardware platforms such as Windows NT, UNIX, LINUX, Oracle, SQL Server, etc. can be allowed, and integration with other applications and data sources using modular design is easily enabled. Through the use of Enterprise Application Integration ("EAI"), the present invention centralizes management of a company's existing software and data. EAI enables message queuing, publish-and-subscribe, XML bridging, as well as leveraging with existing company applications, databases, and systems.

The Enterprise Information Portals ("EIP") functionality of the present invention personalizes and merges information sources. EIP modules access and enable external resources while still providing a user-friendly interface for personalizing information.

The present invention allows content access from PDA, cell phone and other wireless and wire line devices. The present invention enables users to dynamically load stylesheets appropriate to the devices as well as can create mobile user profiles. The system requirements for the present invention are summarized in Table B.

Configuration Option	Minimum Requirements	Recommended
Computer model	Two (2) Sun Enterprise 250 or NT "one for application and one for DB	Two (2) Sun E450 Dual processor or NT dual processor "one for application and one for DB
Dedicated Server'	No	Yes
OS	Sun Solaris 2.7/NT/Linux	Sun Solaris 2.7/NT/Linux
RAM	512 MB	1 GB
Database	Oracle 8.05 and above	Oracle 8i and above
Application Server	WebLogic Application Server	WebLogic Application Server
Disk Subsystem	Non RAID	RAID
Disk space for data storage -	1x current size of Web site	2x current size of Web site
Disk space fair nGia program files	<20 MB	<20 MB
Webserver	Netscape Enterprise or Microsoft IIS *or* Apache	

Table B

At a high level, the present invention has been designed as a 4-tier system. Some benefits of the four tier architecture include, but are not limited to the ability to isolate the effect of code changes thereby increasing the ease of updates and extensibility; to define stable and standardized Web interfaces, to group functions into coherent modules, and to partition work among team members.

Figure 2 illustrates the broad functions of each tier in the present invention's four tier system. In addition, Figure 2 depicts the technologies supporting the function in each tier. The four tiers include the Database Tier 210, the API Tier 212, the Application Tier 214, and the Presentation Tier 216. The API Tier 212 and Database Tier 210 comprise the "backend" of the present invention, and are accessed only indirectly by licensed users via the Presentation Tier 216 and Application Tier 214. The content management toolset that accompanies the software of the present invention is an example of an application developed in the Application Tier 214 that was built upon the API Tier 212. Applications developed using the API set access the Database Tier 210 through the API calls. Site designers work within the Presentation Tier 216 to develop layouts for the information exposed through the Application Tier 214.

The present invention is a method for integrating content and application delivery over a distributed computer network in a single open architecture program. The method comprises entering a URL to a user machine connected to the distributed computer network. Then a prescribed template is retrieved on the received URL. Then the present invention receives over the distributed computer network content and an application. Then the present invention parses the content into a plurality of XML fragments. Each XML fragment is identified by an element type and an element style. The element type indicates a type of XML fragment. The element style indicates a style presentation of the XML fragment. Then the present invention assembles a XML document in accordance with the template. Finally, the present invention transforms the assembled XML document using XSL into a page for display in a browser operating on the user machine. Finally, the present invention displays the page in a browser operating on the user machine.

Other objects and features of the present invention with become apparent from the following detailed description considered in conjunction with the accompanying figures. It is to be understood, however, that the drawings are designed solely for the purposes of illustration and not as a definition of the invention, for which reference should be made to the appended claims.

BRIEF DESCRIPTION OF THE ILLUSTRATIVE EMBODIMENTS

The foregoing and other features of the present invention will be readily apparent from the following detailed description and drawings of illustrative embodiments of the invention wherein like reference numbers refer to similar elements throughout the several views and in which:

FIGURE 1 depicts an overview of the present invention;

FIGURE 2 depicts the four tier architecture of the present invention;

FIGURE 3 depicts the external application functionality of the present invention;

FIGURE 4 depicts an exemplary user login Web page for the present invention;

FIGURE 5 depicts an exemplary Java applet Web page in accordance with the present invention;

FIGURE 6 depicts how a Java applet constructs the content and applications Web pages through the XML template in accordance with the present invention;

FIGURE 7 further depicts the process in Figure 6;

FIGURE 8 depicts attribute selection in accordance with the method of the present invention;

FIGURE 8A depicts an exemplary group edit Web page in accordance with the method of the present invention;

FIGURE 8B depicts an exemplary user edit Web page in accordance with the method of the present invention;

FIGURE 9 depicts the overview of the user administrator method in accordance with the method of the present invention;

FIGURE 10 depicts a further exemplary user edit Web page in accordance with the method of the present invention;

FIGURE 11 depicts a further exemplary group edit Web page in accordance with the method of the present invention;

FIGURE 12 depicts the overview of the site architect method in accordance with the method of the present invention;

FIGURE 12A depicts an elements edit screen;

FIGURE 13 depicts an exemplary element edit Web page in accordance with the method of the present invention;

FIGURE 14 depicts a further exemplary element edit Web page in accordance with the method of the present embodiment;

FIGURE 15 depicts a further exemplary element edit Web page accessed from Figure 14;

FIGURE 16 depicts an exemplary template edit Web page in accordance with the method of the present embodiment;

FIGURE 17 depicts an exemplary XML edit Web page in accordance with the method of the present embodiment;

FIGURE 18 depicts the overview of the site developer method in accordance with the method of the present invention;

FIGURE 18A depicts an exemplary edit Web page utilized by site developers;

Figure 19 depicts a further exemplary site developer edit Web page;

FIGURE 20 depicts a further exemplary site developer Web page;

FIGURE 21 depicts an exemplary Message Handler edit Web page;

FIGURE 22 depicts an exemplary Directory Pool edit Web page;

FIGURE 22A depicts an exemplary Directory Pool edit Web page;

FIGURE 23 depicts an overview of the content manager method in accordance with the method of the present invention;

FIGURE 23A depicts an exemplary content manager edit Web page;

FIGURE 24 depicts an exemplary Deployment parser Web page; and

FIGURE 25 depicts an overview of the parser process;

DETAILED DESCRIPTION OF THE ILLUSTRATIVE EMBODIMENTS

Figure 1 illustrates an overview of the present invention 100. As shown in Figure 1, the present invention 100 supports both content delivery 104 and the technology integration 102 in an open architecture platform 106 that supports data management, business processes, as well as a number of distributed computer network applications.

Figure 3 illustrates the capacity of the present invention to access and present data from many external data sources, through an Extensible Markup Language ("XML") interface 320. Elements of a Web page 324 called functional plug-ins in the diagram are actually mini-applications that retrieve data from databases and external applications, and format the results as XML. A template combines the XML fragments from each Web page element 324 into a single XML document By transforming the XML with Extensible Stylesheet Language ("XSL") 322, content can be presented to a variety of devices in a variety of formats. The elements of the Web page 324 can be applications developed specifically for system using the API set or the elements of the Web page 324 can be calls to existing Enterprise Java Beans ("EJBs") that encapsulate business logic developed outside of the system or from external data sources.

A key feature of a system, such as the present embodiment, that intends to reuse and repurpose Web page content, is the separation of Web page content from Web page layout information. That is, the content should be free of information that specifies how it should look to the user. HTML separate Web page content form Web page layout, because HTML embeds Web page layout within the Web page content. For example, HTML to display a book title could look like this:

```
<b>To Kill a Mockingbird</b>
```

Here the Web page content "To Kill a Mockingbird" is embedded with the Web page layout, bold. With HTML, there is no indication of what the text between the <b> tags signifies; just that it should be emboldened in the browser. The HTML would have to be modified for any device that does not support a bold font, thus preventing scalability (i.e. requiring a new document for every device that may view the content) and introducing unneeded complexity. Furthermore, nothing meaningful can be done with the content outside of the document, because there is no indication of what the content represents outside of a text string. XML avoids these limitations as can be see in the command below.

```
<BookTitle>To Kill a Mockingbird</BookTitle>
```

Instead of indicating the Web page layout of the book title, the XML command indicates that the Web page content "To Kill a Mockingbird" is a book title. By later processing the XML with a XSL stylesheet, the Web page layout will be determined. The XSL stylesheet will describe how to display the book title within a browser or other device accessing the content.

All administrative users (content managers, programmers, etc.) will access the present embodiment through a browser-based Java applet. Individuals visiting an Web site served by the present embodiment will not be required to login unless the Web site is set-up to deliver personalized information and must therefore know the userID. Figure 4 depicts an exemplary user login Web page 400. Once the user has been authenticated, the Java applet of the present embodiment is loaded. Figure 5 depicts an exemplary Java applet Web page 500.

Figure 6 depicts how the Java applet constructs the content and applications Web pages through the XML template. The Java applet depicts both the main control panel 670 and the content frame 672. The two main folders are Docroot, where the content (text and data) is stored, and nGia, where users, groups, templates and elements are accessed.

The main control panel 670 of the Java applet is populated with the structure (directories) of the Web content and application functionality. The main control panel 670 accesses the DirectorySession Bean 674 using Remote Method Invocation ("RMI") to "mount" the entry points into the application and content. The content frame 672 is responsible for data templating. The screen is populated from the PageServlet 676, which calls TemplateSession 678 and associated parsers to load the appropriate XML data.

A Web site served by the present embodiment looks like any other to the end user. Behind the scenes, however, there are many complex operations that result in a Web page. These are the same operations that populate the main control panel 670 of the Java applet interface described above.

Accessing a Web site hosting the present invention triggers the following events depicted in Figure 6.:

The PageServlet 676 parses the request and passes it on to the TemplateSession Java Bean 678. TemplateSession 678 is a stateful session bean – each entity bean instance represents a single template and its elements and parsers.

1. On the way, it accesses the BoundThreadCache Java class to create or access the current user and session information, creating a new thread in the process.

2. The session information is passed to the TemplateSession 678 parse method, which queries the database to determine which elements are associated with the template.

3. The Java embedded classes, the "parsers," 660 associated with each element are executed, returning data wrapped in XML in accordance with the XML schema that defines the element. Data returned can originate from the internal database 664 or any external database or system 664.

4. Elements are populated by the output of the Java classes, becoming in the process "fields" in the Web page. Fields are defined as instances of elements. The data returned by the parser is validated against the XML schema for the element.

5. The results of all parser outputs 660 are aggregated by the TemplateSession 678 into a single XML document. The resulting page can be thought of as an instance of the template. Individual pieces of data (the "content") are stored in the field table of the database, referenced by the pageID.

6. The final output is created by transforming the XML page with the XSL stylesheet that has been associated with the template. The XSL stylesheet itself is actually one of the elements in the template.

Figure 7 further illustrates the initial request depicted and described in Figure 6 to the PageServlet 776. As mentioned above, to create or access the current user and session information a new thread is created, bound (2&3), then the callback of PageServlet 676 with the doProcess method (doProcess calls the parse method of TemplateSession to get elements and parsers as described above).

In summary, PageServlet 676, the main controlling servlet:

- Handles HTTP-based requests and returns HTML (or other transformed) content
- Keeps track of sessions representing unique visitors and associates them with a username.
- Communicates with TemplateSession 678 to determine the appropriate Template to invoke for a given URL.
- Pools references to TemplateSession for optimum performance in a multithreaded context.

In any truly object oriented system, every component is an object with a set of properties. The present embodiment adheres to this standard, and therefore everything in the system is an object with properties. Every object has permissions and actions that can be performed on it. The Java applet interface of the present embodiment embraces the object orientated approach. Each piece of data has an attribute. The user accesses the data attributes with a right mouse button click and with a choice of the appropriate option. For example, selecting "new" with the Groups folder adds a new group whereas selecting "new" with the Templates folder creates a new template.

Figure 8 illustrates the pull down menu 896 illustrating the attributes that would appear with a right mouse button click or any other actuating action. As can be seen in Figure 8, the user has actuated the nGia folder. The attribute highlighted in that attributes list pull down menu 896 is as can be seen the "new" attribute.

Before interacting with the present invention, user and groups must be set up. An example of a group edit screen can be seen in Figure 8A. Groups are added to the system for security and access purposes. Users can be in multiple groups and can have different roles within different groups. As shown in Figure 8A, there is a user's list 840 and a group list 842 In addition, the present invention enables groups to be nested within one another with the groups within groups list 846. The users and groups to which they belong, along with objects and permissions comprise the security model of the present embodiment.

The security model is based on users and groups who have permissions on objects. Users must belong to a group, and the group in turn has permissions on objects. There are 3 axes which must be considered when determining the rights on an object: the object itself, the group, and the permissions.

Some examples of valid permissions include, but are not limited to read, write, create, and destroy. Each object created in the system has an owner, and only the owner can change the permissions of the object. When a new object is created, it is given all permissions for its owner group, and no permissions for anyone else. Thus members of that group will inherit the capability of changing the permissions on an object it owns. For example, if the Content Manager group is set-up as the principle that owns all new pages added to the Web site, then any member of that group may change permissions on a new page, provided that that user is not in a subgroup whose write permission has been revoked for that object.

Users do not have individual permissions; groups do. Users are individuals having access to various functions within the system. User access rights are associated with the groups to which a user belongs. A group can have one or more users, and up to 3 levels of nested groups (groups within groups). When a user is added to the system, the user is automatically assigned to the group Everyone, which has the lowest level of permissions. Users accessing a Web site hosting the present embodiment are considered members of the group Everyone and assigned permissions accordingly. An example of a users edit screen 848 can be seen in Figure 8B.

USER ROLES

What follows is a listing of the standard roles that users have within the present invention. These are the typical users who transact with the system of the present embodiment. The following roles define the present invention's core use cases:

- User Administrator

- Site Architect
- Site Developer
- Site Designer
- Content Manager
- Message Administrator
- Directory Administrator

For most of the cases listed above, a general process flow diagram will be presented showing the options each has, as well as the actual Web interface screens they would use to perform their tasks.

USER ADMINISTRATOR

The user administrator is responsible for setting up and administering the users and groups. The process flow for user administrators in the present invention is depicted in Figure 9. As can be seen in Figure 9, the objects the user administrator can access are the users and groups objects.

User administrators add and modify users of the system using the user edit screen 1048 shown in Figure 10. Users are objects like everything in the system, and have permissions associated with them as well. When right-clicking on the users folder and selecting "new", the user edit screen 1048 appears with blank data fields, provided the user administrator has permissions to add another user. When opening the user folder, selecting a user (in this case

GREENROCKET), and selecting "edit" from the pull down menu, the user edit screen results 1048 with populated data fields for editing.

Groups are set-up similarly using an groups edit Web page as shown in Figure 11. On this screen users are placed into the selected group, and groups can also be placed within the selected group, if desired. In addition, as discussed above users can be in more than one group, and have different roles within different groups. There are 9 predefined groups with already set permissions: Everyone, Content Manager, Directory Administrator, Message Administrator, Site Architect, Site Developer, User Administrator, Super Group, and System. Their permissions are summarized in the Table C.

Group Name	Member Groups	Special Properties/Notes
ContentManager	UserAdministrator, SiteDeveloper, SiteArchitect	Logs into admin screens (otherwise they will be refused); Adds and edits web pages; Uploads files; Reviews and approves Web page changes
SiteArchitect	SuperUser	Creates site structure by building templates and elements
SiteDeveloper	SuperUser	Creates new embedded classes; Develops functional components; Integrates legacy applications and data
SiteDesigners		Creates and edits page layouts Uploads graphics and multi-media
UserAdministrator	SuperUser	Creates new Principles (both users and groups)
MessageAdministrator	SuperUser	Adds/removes MessageHandlers
DirectoryAdministrator	SuperUser	Adds/removes/edits DirectoryPool mounts
SuperGroup		Is given all permissions to all

Group Name	Member Groups	Special Properties/Notes
		objects, either hardcoded into app logic, or whenever new rows are added
Everyone/Guest		Default permissions for unauthenticated users, such as typical web browsers
SYSTEM	None	Contains only one User, GREENROCKET. The SYSTEM group is invisible to everyone, except DMIND developers, and owns exclusive rights to all the reserved groups and system objects.

TABLE C

SITE ARCHITECT

The site architect develops the overall information architecture of the Web site, including defining the needed elements and templates. Elements are created with XML schema and associated with a parser for data population. Templates are built from groups of elements. The goal of the site architect is to ensure that any repetitive or structured information is normalized and ordered in a consistent hierarchical manner throughout the entire Web site.

Figure 12 depicts the overview of the site architect method in accordance with the method of the present invention. As shown in Figure 12, the site architect has access to the objects "elements," "template," and "mount points." One purpose of the XSL in the system of the present embodiment is to dictate how XML information generated by the system should be displayed on a Web page. For example, the XSL may indicate that the element "employeename" should be in bold font and centered. It is up to the site architect to determine the site structure and data element characteristics. The site architect defines elements by inputting XML schema. The site architect then associates these XML schema with parsers developed by the business logic programmers. Finally, the site architect constructs templates out of these elements.

The site architect role in conjunction with the other user roles compartmentalize the responsibilities of each person using the system of the present embodiment. The system then enables the underlying business structures to be handled by a single role who is responsible for modeling the core business structure. Underlying business structures are abstract enough to be free of design or content specifics. Consequently, the system takes advantage of the abstracted business structures. The system normalizes and orders any repetitive or structured information in a consistent hierarchal manner throughout the entire Web site.

ELEMENTS

Elements are the basic building blocks of the system, representing the information that will be shown on the Web pages built by the system. Each element supplies a Document Type Definition ("DTD").

The content section of the present invention is generated dynamically, based on the DTD fragments of the elements. The application can obtain the DTD for an element from its DTD body. The following restrictions apply to the element's DTD:

- The Element must have one <!ELEMENT> tag matching its name.

- This ELEMENT tag must include an attribute list of fixed attributes:

- The element's ELEMENT tag cannot include any other attribute list other than the one described above.

- At least one user-defined child element must be included in the element's child list.

- The reserved element exception must be included as one of the element's optional children, unless it is a basic DTD section.

- With the exception of an attribute list for the element's ELEMENT tag, any other markup, including other ELEMENT declarations, may follow.

An example of an elements edit screen can be seen in Figure 12A. Elements are represented by XML schema, such as:

```
<?xml version="1.0" encoding="UTF-8"?>

<schema>

<element name="Content" type="string">

<annotation>
```

```
<documentation>Could not update this content because it is not valid for the
specified data type.</documentation>
        </annotation>
        </element>
        </schema>
```

The schema is used to indicate the XML that will be returned by the parser 660, to categorize it (in this case as a "string" with the XML tag "Content"). The Page Servlet 676 compares the output that is expected (indicated by the schema) with the output returned in order to validate it.

The element name in the schema ("Content") is a special XML tag used by the Content parser 660, to wrap incoming data. It can be used for many different elements, thus this name is different from the name that the schema is stored under in the database. For example, elements named FirstName and LastName may be created by a user, with the schema element name "Content"

The present invention automatically returns a certain amount of XML to describe the Page ELEMENT, the first XML ELEMENT of every document. It then works with the Template Element. It also inserts the ELEMENT tag for each element, calls its parser, then closes its ELEMENT tag. Note that the "meta info" ATTLISTs for each element will be automatically populated, since they have FIXED values.

The XML returned by the parser would then look like this for the schema indicated above for an element named "FirstName":

```
<ngia:FirstName>
        <ngia:Content>Robert</ngia:Content>
</ngia:FirstName>
```

The "ngia:" prefix to the element name is the namespace for that element. Namespaces must be used with XML schema to distinguish them from elements of the same name created elsewhere. Thus namespaces reduce the chance of naming conflicts among elements. Namespaces are manipulated on the screen shown below. Each prefix (such as "ngia") is mapped to a fully qualified URL. New namespaces can be added here, and existing ones deleted. Figure 13 depicts the element namespace process. As can be seen in Figure 13, each namespace 1330 has a prefix and a URL. For instance as shown in Figure 13, one prefix is "test1" having the URL http://www.test.com

It should be noted that present invention also defines a reserved processing instruction for importing DTD syntax from another Element. This Element's DTD fragment will simply be appended to the end of the Template DTD, apparently allowing its Elements, Attributes, and Entities to be accessed from within side the definition of another Element.

Site architects are responsible for creating the elements as shown in Figure 13 and they work with the client and the content itself to determine the best way of breaking it down, or representing it through the element structure. Content can be populated in the system either through the element mechanism, or through the XSL stylesheet associated with a template. For example, the user may wish to represent the company's privacy policy as static content within the

XSL stylesheet that is loaded each time a page is created from that template, or as a PrivacyPolicy element that can be modified from a Web interface. The element versus XSL stylesheet decision will determine whether or not end users will be able to serve as content managers. XSL stylesheets are typically edited only by users with a higher level of permissions within the system than content managers.

The present embodiment supports all sorts of content. For instance, the present embodiment supports Unicode character encoding, two-byte Asian characters, and localization in any language.

The present embodiment supports two types of elements, namely, page elements and template elements. A page element is one which changes for every page that is generated from a template. Page elements originate from more dynamic, changeable content. Template elements remain constant for every page based on a template. They are thus based on more static content. An example of template elements are the editor and viewer XSL stylesheets because they are not page specific.

Figure 14 depicts an the element edit Web page in accordance with the method of the present embodiment. As shown in Figure 14, a site architect enters the XML schema for the element 1434. The present invention provides a convenient mechanism to create a "skeletal" schema with valid XML as a starting point for the user. The appropriate field type for the element (page or template) 1432 is also selected here, and default parameters 1436 are set as well. The default parameters option 1436 provides a way to pass arguments to the parser that are specific to the element.

Figure 15 depicts a further element edit Web page accessed from Figure 14. Figure 15 highlights the default parameter option 1436. As can be seen in Figure 15, the user has selected the "User Defined" XML schema 1534 for the element. Now, the user must choose the default parameters 1536. As mentioned above the default parameters option 1536 provides a way for the user to pass arguments to the parser that are specific to the element. One use of default parameters option 1536 would be to choose a subset of data from the parser, or tell the parser which directory to perform its operation (e.g. "directory=/docroot"). The appropriate parser is associated with the element by selecting it from the default parameters pull down menu 1536 as shown directly above. The XML for the element is stored in the database upon submit, and retrieved any time the element is triggered by the template in order to validate the output of the associated parser.

TEMPLATES

Templates are built by selecting elements from a library and adding them to the template. Figure 16 depicts a exemplary template creation Web page. The template is thus a collection of XML schema, and represents fully the data that will make up any page based on that template. As can be seen in Figure 16, the system presents to the user both a list of elements 1538 and a list of template elements 1550. To reiterate, elements will produce data either from information stored in the internal database, or from an action that gets information from an application or external database. The template is a Web page representation of the data (content) elements present in this Web page. For example, on a human resources page there may be elements for employee name, social security number, and address that come from the HR database, as well as an element that pulls information from a news feed.

SITE DESIGNER

The site designer implements the look and feel of the Web site using XSL stylesheets. This may be the same person as the site architect. There are two "hard coded" XSL stylesheets with every template, one which creates HTML for the editor of the page (a template element called EditXSL), and one which creates HTML for the viewer of the page (a template element called ViewXSL). However, any number of XSL stylesheets may be used at runtime to display the final output, by including the appropriate argument in the URL. The XSL for the hard coded XSL stylesheets is entered into the system using an XSL interface 1552 such as displayed in Figure 17, and upon "submit" this information is stored in the database and associated with the template. The purpose of the XSL is to dictate how the XML information generated by the system should be presented (laid out) on the page. For example, it may indicate that the element called EmployeeName should be in a bold font and centered.

The XSL references the elements that have been associated with the template. The majority of the business logic that determines which elements can be viewed by which users is implemented at the parser level, in order to keep the XSL stylesheet as simple as possible. An excerpt of a sample XSL stylesheet is shown below:

```
<?xml version="1.0" encoding="UTF-16"?>

<xsl:stylesheet version="1.0"

        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        xmlns:ngia="http://www.dmind.com/ngia/application"
                xmlns:uni="http://www.unispherenetworks.com"
                xmlns:sys="http://www.dmind.com/ngia/system">
<xsl:output

        method="html"
        omit-xml-declaration="yes"
                indent="no"
                />


<xsl:template match="sys:Document/sys:Content">

<!-- Copyright 2000 DMind Corporation -->

<!-- URL: http:/www.dmind.com -->

<html>

<head>
```

```
                <meta http-equiv="content-type" content="text/html;charset=euc-kr"
/>

                <title>Unisphere Solutions - News</title>

                <style type="text/css">

                 ...

                <h3>Press Releases</h3>

 <xsl:if test="uni:ContactName/uni:Content and uni:ContactName/uni:Content!=''">
                <xsl:value-of select="uni:ContactName/uni:Content" disable-output-
escaping="yes"/><br />
</xsl:if>

  <xsl:if test="uni:ContactTitle/uni:Content and uni:ContactTitle/uni:Content!=''">
                <xsl:value-of select="uni:ContactTitle/uni:Content" disable-output-
escaping="yes"/>

                <br />
```

As illustrated above the XSL stylesheet is primarily HTML with embedded references to the XML elements associated with the template. Although XSL is capable of performing logical and conditional operations, the present embodiment encourages simple XSL stylesheets, with complexity relegated to the parser level.

SITE DEVELOPER

The site developer develops Java parsers, also known as embedded classes, that deliver data from internal and/or external sources. Many of the parser requirements will come from needs defined by the site developer and client. Figure 18 depicts an overview of the site developer method in accordance with the method of the present embodiment. As shown in Figure 18, the objects available to the site developer include access to modules such as DirectoryPools, Embedded Classes, and MessageHandlers.

An exemplary of a edit Web page utilized by site developers is shown in Figure 18A. With the web page depicted in Figure 18A, site developers add their own Java application code 1854 to the system. Figure 18A is known as an embedded class edit screen. Embedded class files are compiled by the system and inserted into the database as executable byte-code. When an element is called that is associated with one of these parsers, the system executes the element and returns data in accordance with the element's specification. In this way, the present embodiment enables the addition of custom applications and/or classes that invoke an external Enterprise Java Bean.

PARSERS

Parsers are business logic modules that encapsulate CRUD (Create, Read, Update, Delete) actions for content fragments. Whereas elements as discussed above return DTD fragments, parsers return XML fragments. For better parsing performance, the user receives a standalone XML document containing both DTD and XML fragments. One Page can activate an arbitrary number of Parsers to handle its business logic, and these Parsers allow the present invention to manage content, on a granular level, from many data sources.

Parsers are responsible for representing requested data in XML, and this data is in turn verified by criteria defined in the XML schema for the associated element. Parsers format data in XML's display neutral syntax, promoting the reuse of parsers between many presentation scenarios. Parsers extend an Abstract Class that dictates their functionality, and are deployed and loaded through the module platform of the present invention.

The present invention provides the facility for uploading an existing compiled Java class with the OS based Browse window 1990 as depicted in Figure 19. Figure 19 depicts a further exemplary site developer edit Web page. Programmers specify the class run type as a Caller 1982 or Principal 1984, meaning whether the module runs as its invoker (caller) or as its own unique identity.

As described earlier, the applications written for the present invention and used as parsers are accessed by the system through one or more elements in a template. When an element is called that is associated with one of these parsers, it is executed and returns data and verifies it against the element's XML schema. This is a way to extend beyond the out-of-the-box functionality of the present invention by adding custom applications and/or classes that invoke an external Enterprise Java Bean or data source/application.

It is expected that site developers write and compile the applications externally to the present invention, then upload the compiled class files 1994 into the system. Figure 20 depicts a further exemplary site developer Web page illustrating this upload functionality. Figure 20 depicts an exemplary upload file screen 2092. Using the upload file screen, the compiled class 2094 is then inserted into the database as executable byte-code, at which point it becomes available for associating with an element.

Parsers, Message Handlers and Directory Pools are all modules within the present invention. Modules are Java classes that extend or implement Java interfaces or Java abstract classes. The byte-code for a module is stored in a database table along with certain metadata.

MODULE PLATFORM

When a module is needed by the application, it uses a custom class loader that extends the basic class loader provided by Sun with the Java Development Kit. Instead of searching for the class on the file system, which is the functionality supplied by Sun, the present invention's class loader examines the appropriate database table and loads the respective byte-code. Using this technique, class libraries are centralized in a single repository that provides greater availability in a distributed application environment. All modules (parsers, message handlers and directory pools) utilize this mechanism.

MESSAGE AND DIRECTORY ADMINISTRATORS

Message Administrators and Directory Administrators are also programmers, and they use the same screen to upload their compiled code. While not shown in the screenshot above, programmers will be able to indicate which type of module their compiled class represents, a message handler, directory pool, or embedded class (parser). Message Handlers are modules that respond to specific application events, such as viewing a certain element on a page, adding a page to the system, or a request for a page by an unauthorized user.

Message Handlers are built on top of the Java Messaging Service, a vendor-independent message oriented middleware ("MOM") API. Message Handlers accelerate the development time of MOM modules that serve as custom adapters to external services, such as third-party versioning, deployment scripts to specialized production environments, traffic analysis/personalization packages, and synchronization with external user repositories. Message Handlers also provide a convenient way to catch application events and send out email notifications, or build business-specific workflow processes.

When the compiled code for a Message Handler has been inserted into the system, it can then be associated with a "topic." Figure 21 depicts an exemplary Message Handler edit Web Page. The topic 2156 shown in Figure 21 dictates the event or condition that the Message Handler responds to. The Message Handler continuously listens to the system for the events 2156 and performs an action with the listen condition is met. For example, one listen condition 2156 can be a web page modification or a date. The action could be an email notification. The system sends notification to the message handler whenever the specified date passes or web page has been modified. Upon receiving notification that the condition has been met, the message handler would then send an e-mail to an appropriate individual informing them of the event.

DirectoryPools, which are developed by users, partners, or clients of the present invention, may serve as a façade for third-party applications, enabling external data sources to serve as data models for sections of the nGia directory tree. They essentially provide a "mount point" for existing external directories or software services, so that they can be accessed via the system's web interface. An example of a Directory Pool edit screens can be found in Figures 22 and 22A. As shown in Figure 22, the user is editing the DirectoryPool "Pool4."

A DirectoryPool is Java class that may be deployed at runtime to represent a specific section of the Web site, known as a mount point. In Figure 22A, the mount point 2286 is depicted. The DirectoryPool mapped to this mount point 2286 establishes a common set of policies for finding, creating, removing, and searching within this mount point 2286. Clients obtain directory information through a single application entity, the DirectorySession EJB. This object delegates directory requests to a concrete DirectoryPool implementation according to a reserved algorithm. This algorithm analyzes the incoming request and compares it to its list of mount points 2286, maintained in memory. If the full path of the request has no match, it trims off the last token of the request and tries again, keeping track of all the trimmed tokens. Eventually, the DirectorySession object will resolve the request, worst case to the DocrootPool, whose mount point 2286 is the virtual document root. The trimmed tokens for a relative request which is forwarded to the matching DirectoryPool, then resolves the request according to its own business logic, perhaps by querying a third party application or database. A potential use of this function would be to make assets that already exist in a file system accessible via a folder in an applet of the present embodiment.

Figure 22A depicts the Web interface for users to mount Directory Pools. Figure 22A depicts the DocrootPool as one of many potential Directory Pools 2288 in the browser pull down window 2296. In keeping with the example given, a pool could be written that accesses a specific directory in a file system and reads the contents, making them accessible via the folder frame 2270 of the applet.

CONTENT MANAGER

The content manager is empowered to make changes to content within the system. Access is strictly controlled to individual elements and pages within the Web site. The template, and individual element permissions, control the area of influence the content manager has. In addition, the workflow component assures that any changes go through a process of approval and feedback.

Once the look and feel and functionality have been implemented in present embodiment, the content can be added to the new templates, creating pages within the system. The content contributor's responsibilities include populating the initial content, and updating it after deployment. An overview of the content manager's role in the present embodiment is depicted in Figure 23. As shown in Figure 23, the objects available to a content manager are web pages and file names. The extent of the content manager's ability to alter Web pages is determined by the content manager's permissions.

Figure 23A depicts two exemplary content manager edit Web pages 2300. In operation, content managers would first navigate to the web page they intent to edit. The web page to be edited 2372 is shown in the frame on the right hand side of Figure 23A Once at the web page 2372, content editors choose EDIT from the pull down menu 2396 in the lower left hand frame of the web page 2300. Content editors will then be presented with a screen 2358 which allows the content editor to edit the web page in the right hand frame. Depending on the template being edited, the content added or modified will shown in the context of the actual page. Others templates will be rm-based and will consequently dictate an input page without the actual page.

CONTENT DELIVERY

Web sites created in accordance with the method of the present invention are most commonly served live from the database, to take advantage of the personalization and dynamic data capabilities. However, in some cases it may be necessary to create static versions of the Web site, to serve from multiple distributed Web sites that do not have the application or database installed of the present embodiment. To provide this facility the Deployment parser was written which writes out the entire Web site into static HTML files, stored in the file system. The Deployment utility depicted in Figure 24 allows the user to specify the root directory (origin of the content), starting point for the deployment, and target directory for file storage.

In sum, the present invention extends existing cutting-edge technologies in a way so as to provide capabilities not presently available in the marketplace. Some benefits of the present invention include a purely object based architecture, a unique module deployment platform, a new templating architecture, a unique framework for managing and integrating external data as well as an extensible event-driven framework.

Thus, the invention can be understood as including several discrete components or layers that cooperate with one another, including:

Presentation Tier:

1. Screens

- HTML-based screens representing viewing and updating all first-class objects.
- JavaScript based form checking
- All screens transformed into HTML using XSL

DirectoryApplet

- Occupies left frame of application real estate.
- Displays directory structures in two panes, one for directories, one for files.
- Allows easy Web site navigation, and dropdown menu for common tasks. Most actions populate right frame with an HTML-based form appropriate to the request.
- Uses RMI connection directly to DirectorySession.

2. Gateway:

PageServlet

- Handles HTTP-based requests and returns HTML-based content.
- Keeps track of sessions representing unique visitors and associates them with an username.

- Communicates with DirectorySession to determine the appropriate Template to invoke for a given URL.
- Pools references to TemplateSessions for optimum performance in a multithreaded context.

3. Application Tier:

- About eight pools provisioning all first-class objects
- Parsers provisioning first-class objects and building XML according to system Elements and Templates
- Workflow parsers and MessageHandlers for generic page staging and approval system

4. API –Session Objects:

DirectorySession

- Stateless session bean, meaning one DirectorySession bean instance is equivalent to any other instance.
- Handles all directory-related requests, including page creation, deletion, directory creation, moving pages, renaming, and determining the template of a page.
- Keeps a mapping of mount points and directory pools. Incoming requests are delegated to the appropriate directory pool based on its mount points.

TemplateSession

- Stateful session bean – each bean instance represents a single template and its elements and parsers.
- Handles all requests for building pages.
- Pages are built out into XML by delegating to all Parsers associated with the Template.
- Page fields, meaning the raw data that each parser will use, is loaded by TemplateSession before delegation.

SecuritySession

- Handles authentication requests, and user/group/permission queries.

VersionSession and SnapshotSession

- Stateless objects for creating versions and snapshots
- authorized users can "rollback to earlier versions of content and Web site updates are queued in a user-defined workflow process;
- Utilizes existing business process approval cycles for web Web site management (e.g., changes approved by supervisors in a way that mirrors the organizational structure)

5. API –Entity Objects:

ElementEntity- Represents a single Element. Properties include

- Schema – XML Schema of the Element. Defines the data type of the Element.
- isStatic – whether or not the Element's fields are identical between all pages of a template or vary between in page
- Parser – the parser that builds the data for the element and handles its updates
- Default parameters – parameters passed into the Element's parser every time it is invoked

TemplateEntity – represents a single Template. Properties include

- Elements – the elements included in this Template

NativeFieldEntity – represents a Field of an Element that is housed in the database. Properties include

- Content – body of the field
- Path – page path of the field
- Element – element parent

EmbeddedClassEntity – represents an module; a Java class file housed in the database. Properties include

- run mode – whether the module runs as its invoker (caller) or as its own unique identity
- Byte-code – actual code of the class

GroupEntity

- name
- group members

UserEntity

- name
- email
- password

6. API – Messaging:

HandlerRegistry – a standalone Java program that runs on the Application Server and subscribes to reserved JMS topics, thus listening to all application-generated events.

- Obtains all MessageHandlers from the database at startup

- Delegates messages to appropriate MessageHandlers as events transpire
- Automatically registers and activates new MessageHandlers as they are created

7. Adapter Tier:

ResourceAdapter

- Loads all drivers at run-time based on configuration files
- EJBs and application modules use ResourceAdapter to determine the appropriate drivers programmatically
- courtesy methods for bean lookup, JNDI authentication, obtaining database drivers, obtaining XML drivers, etc

As explained, the present invention's flexibility lies in its ability to be easily extended and integrated with other systems and data sources. In order to facilitate this extensibility, The present invention allows "snap-in" Java components that contain custom business logic to be synthesized into the system on the Application tier. The present invention includes the following tiers: Database, Adapter, API, Application, and Presentation.

There are three types of "snap-in" components, or embedded classes that the present invention can be extended with:

- Parsers
- Directory Pools

Message Handlers

Each of the components listed above are java classes that can be uploaded and registered with the present invention and extend the present invention's functionality seamlessly. Parsers carry out specific business logic and return XML-wrapped data to the client. A parser is associated with any number of Elements in the system and any number of Elements may be associated with any number of Templates. Figure 25 depicts the parser process. As shown in Figure 25, when a page is requested from the system, the present embodiment executes each parser associated with the relevant Template and gathers their XML fragments to assemble one master XML Document that is consumed by the client and then transformed (using XSLT) into an appropriate format.

WHAT IS CLAIMED

1. A method for integrating content and application delivery over a distributed computer network in a single open architecture program, comprising:

(a) entering a URL to a user machine connected to the distributed computer network;

(b) retrieving a prescribed template on the basis of the received URL;

(c) receiving over the distributed computer network content and an application;

(d) parsing the content into a plurality of XML fragments, each XML fragment identified by an element type and an element style, the element type indicating a type of XML fragment, the element style indicating a style presentation of the XML fragment;

(e) assembling a XML document in accordance with the template;

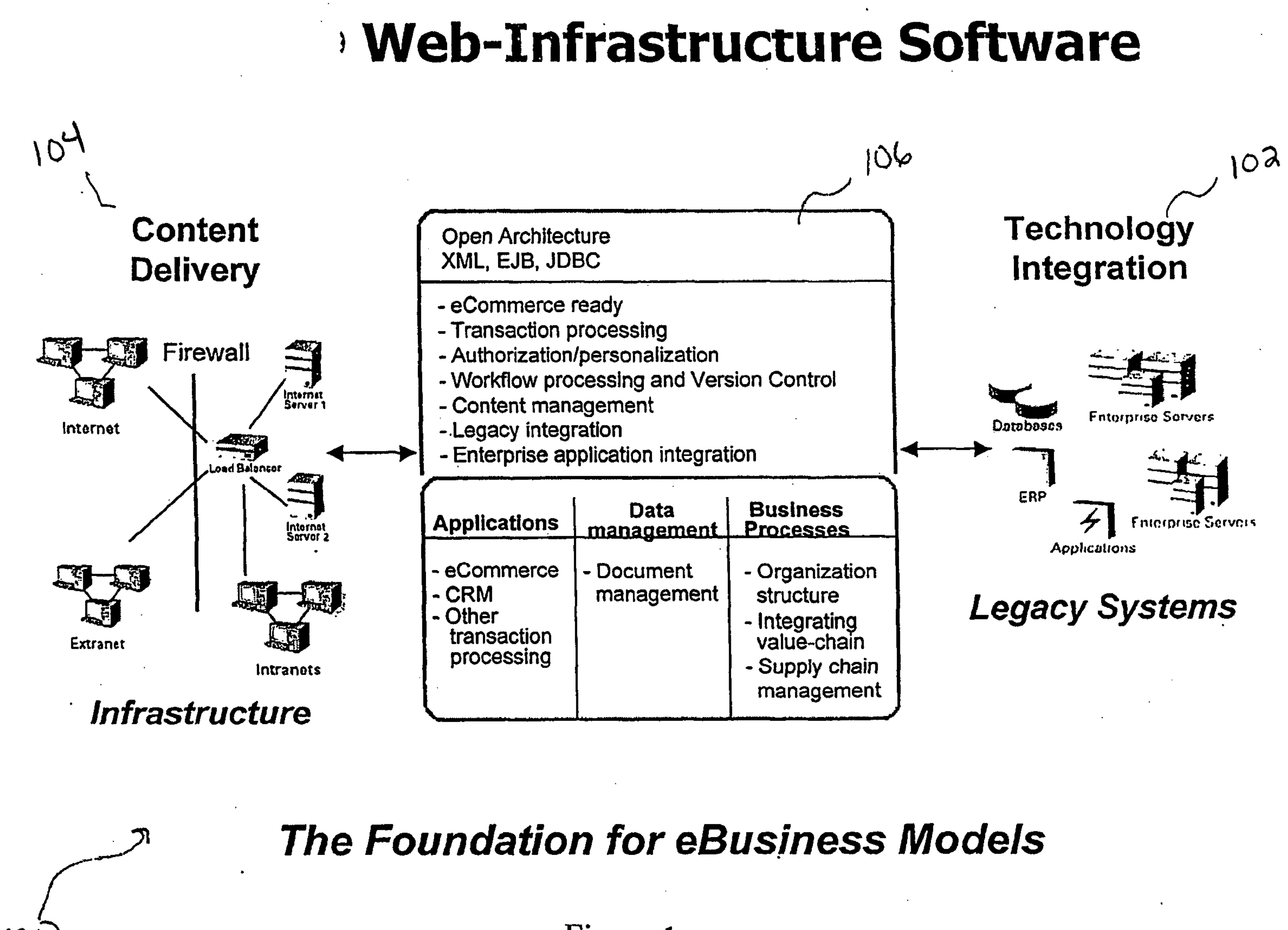
(f) transforming the assembled XML document using XSL into a page for display in a browser operating on the user machine; and

(g) displaying the page in a browser operating on the user machine.

2. A method as in claim 1, wherein the XSL uses the element style to determine a presentation of the XML fragment on the page.

3. A method as in claim 1, wherein the assembling step further comprises integrating the application into the page.

4. A method as in claim 1, wherein the prescribed template for the user is selected in accordance with data in a permissions list.

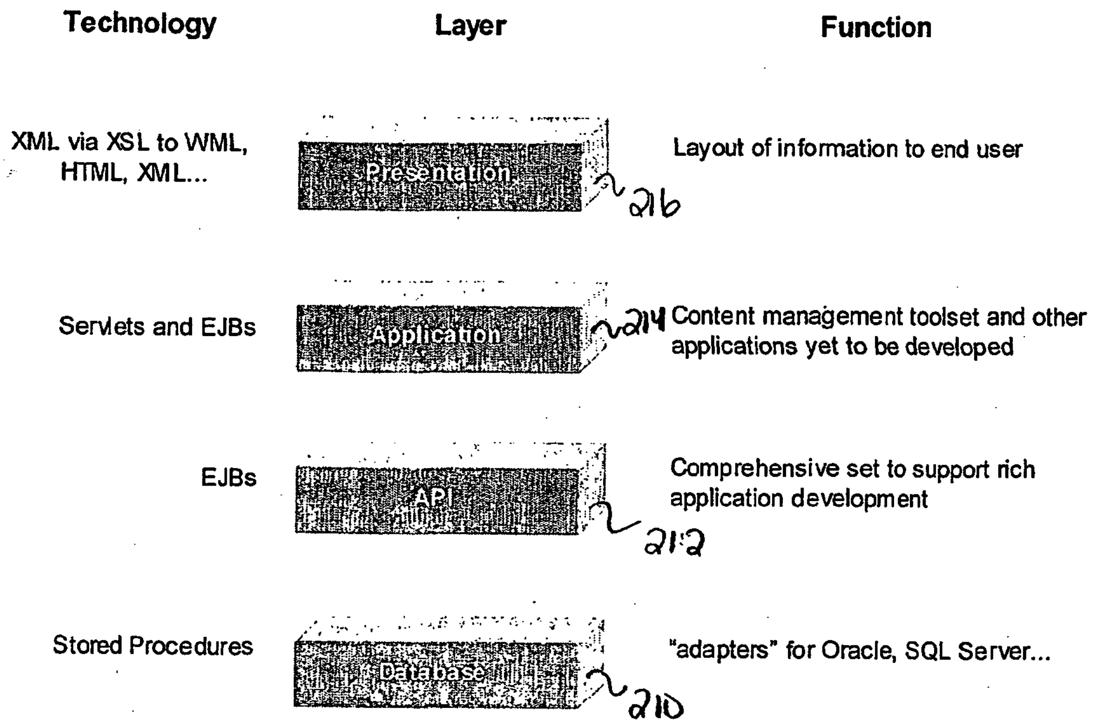


Open Architecture
XML, EJB, JDBC

- eCommerce ready
- Transaction processing
- Authorization/personalization
- Workflow processing and Version Control
- Content management
- Legacy integration
- Enterprise application integration

Applications	Data management	Business Processes
- eCommerce - CRM - Other transaction processing	- Document management	- Organization structure - Integrating value-chain - Supply chain management

Figure 1



Technology	Layer	Function
XML via XSL to WML, HTML, XML...	Presentation	Layout of information to end user
Servlets and EJBs	Application	Content management toolset and other applications yet to be developed
EJBs	API	Comprehensive set to support rich application development
Stored Procedures	Database	"adapters" for Oracle, SQL Server...

Figure 2

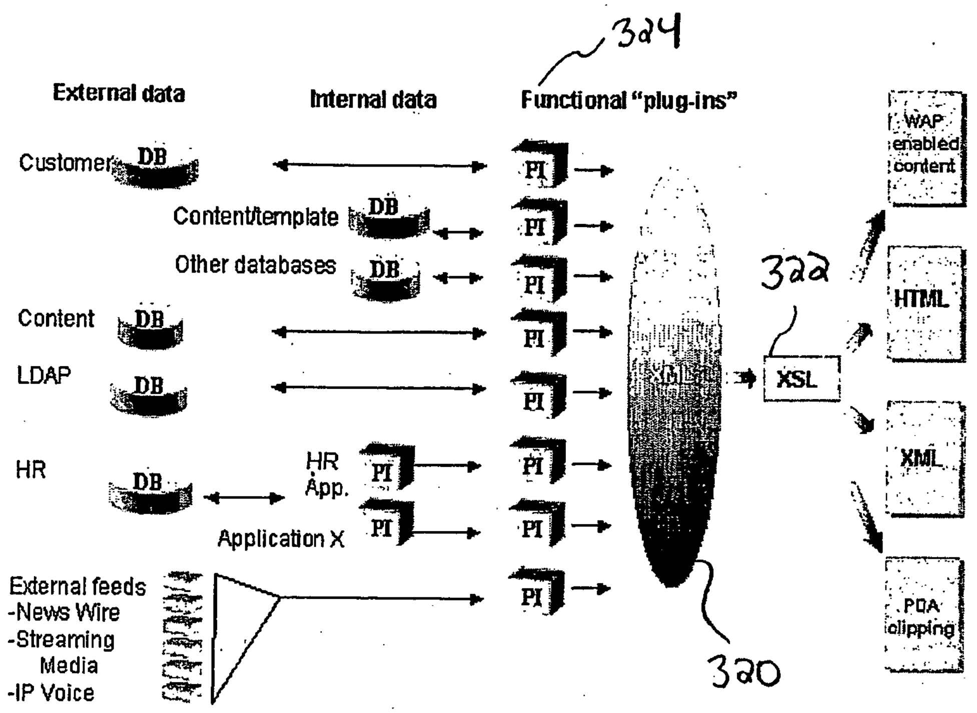
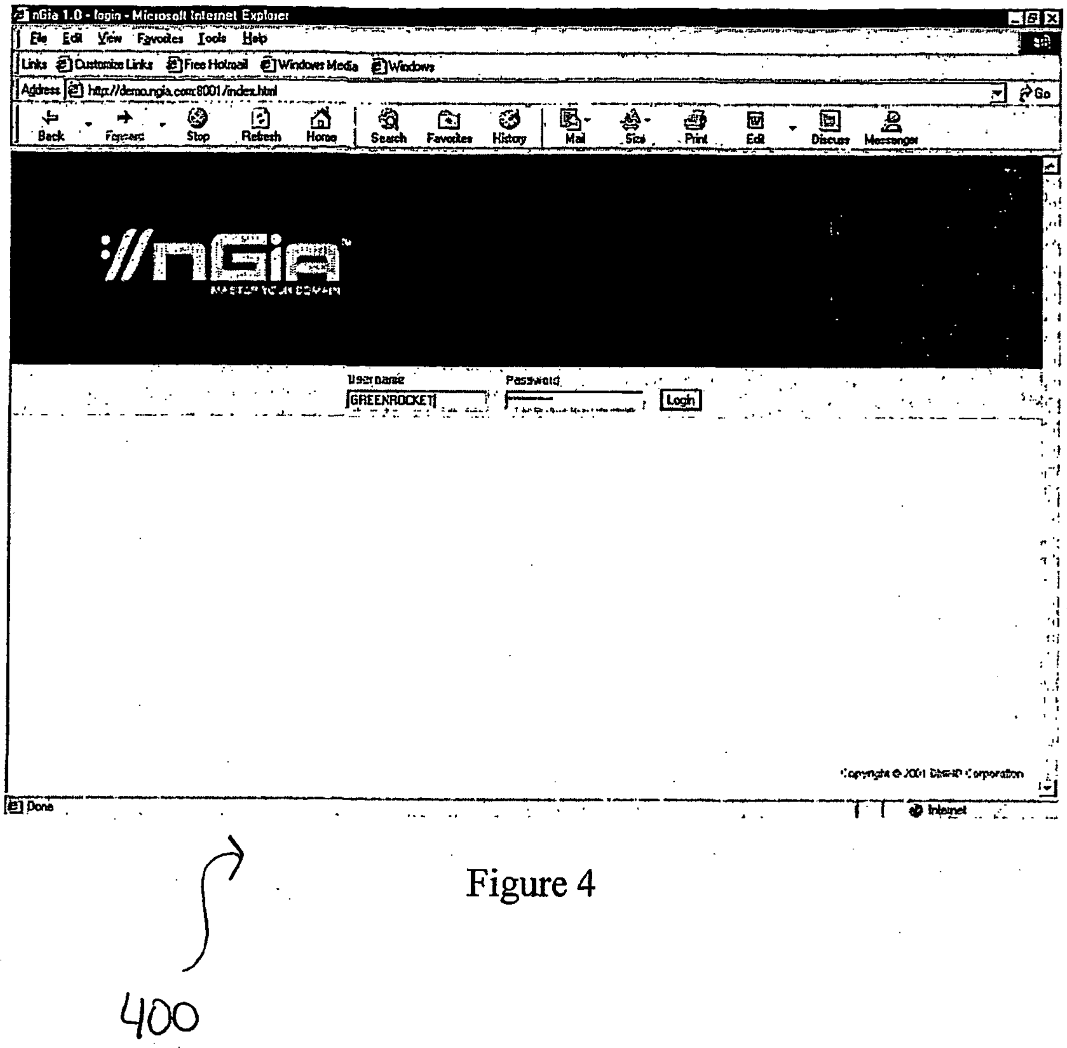

324
External data
Internal data
Functional "plug-ins"
Customer
DB
Content/template
DB
Other databases
DB
Content
DB
LDAP
DB
HR
DB
HR App.
PI
Application X
PI
External feeds
-News Wire
-Streaming Media
-IP Voice
PI
PI
PI
PI
PI
PI
PI
PI
XML
322
XSL
WAP enabled content
HTML
XML
PDA clipping
320

Figure 3



nGia 1.0 - login - Microsoft Internet Explorer

Address http://demo.ngia.com:8001/index.html

Username GREENROCKET

Password

Login

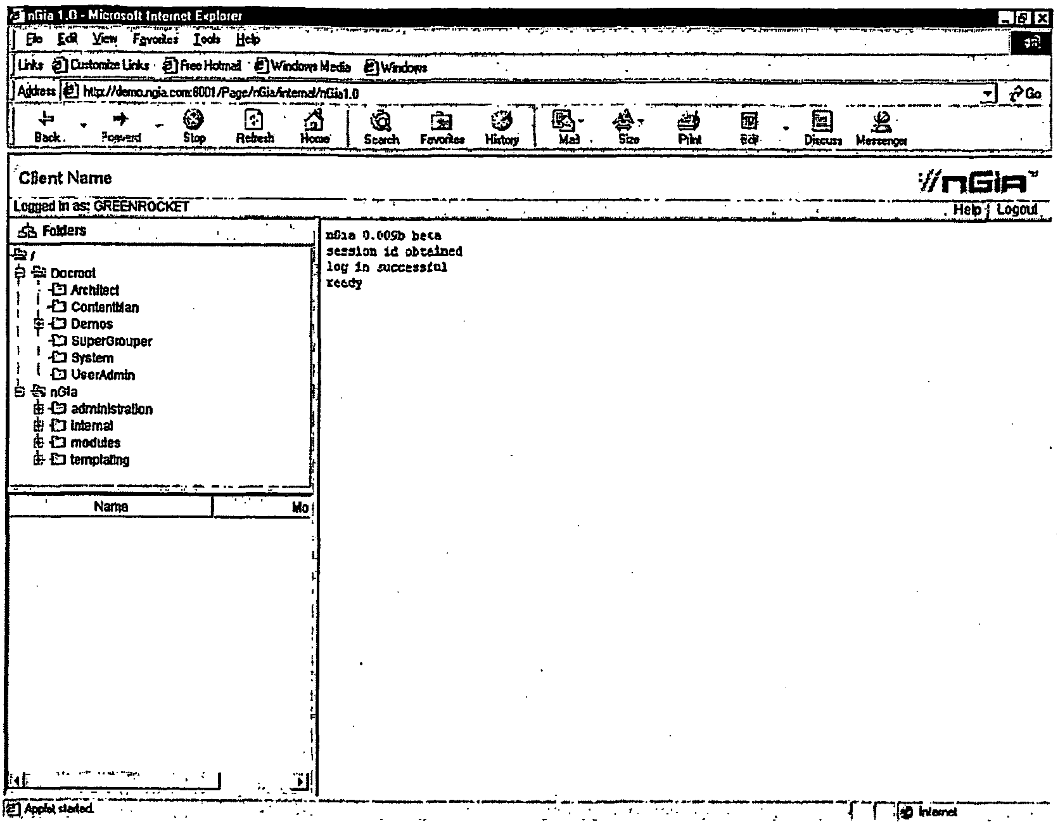
Copyright © 2001 [illegible] Corporation

Done

Internet

File Edit View Favorites Tools Help
Links Customize Links Free Hotmail Windows Media Windows
Go
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss Messenger
//nGiA
400

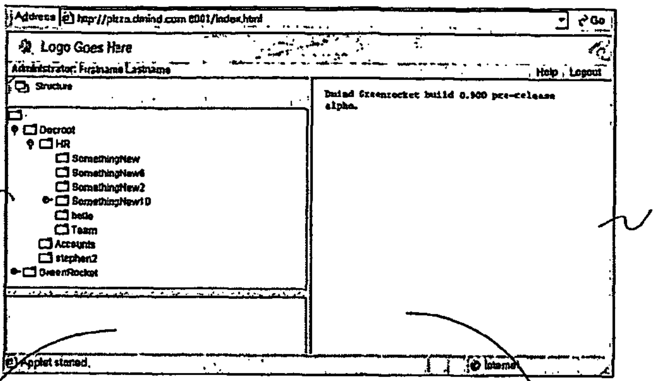
Figure 4


nGia 1.0 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links Customize Links Free Hotmail Windows Media Windows
Address http://demo.ngia.com:8001/Page/nGia/internal/nGia1.0
Go
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss Messenger
Client Name
//nGia
Logged in as: GREENROCKET
Help Logout
Folders
/
Docroot
Architect
ContentMan
Demos
SuperGrouper
System
UserAdmin
nGia
administration
internal
modules
templating
Name
Mo
Applet started.
Internet

```
nGia 0.005b beta
session id obtained
log in successful
ready
```

Figure 5

500



http://plzza.dmind.com 8001/index.html

Logo Goes Here

Administrator: Firstname Lastname

- Decroot
  - HR
    - SomethingNew
    - SomethingNew6
    - SomethingNew2
    - SomethingNew10
    - hello
    - Team
  - Accounts
  - stephen2
- GreenRocket

```
Dmind Greenrocket build 0.900 pre-release
alpha.
```

Applet started.

Internet

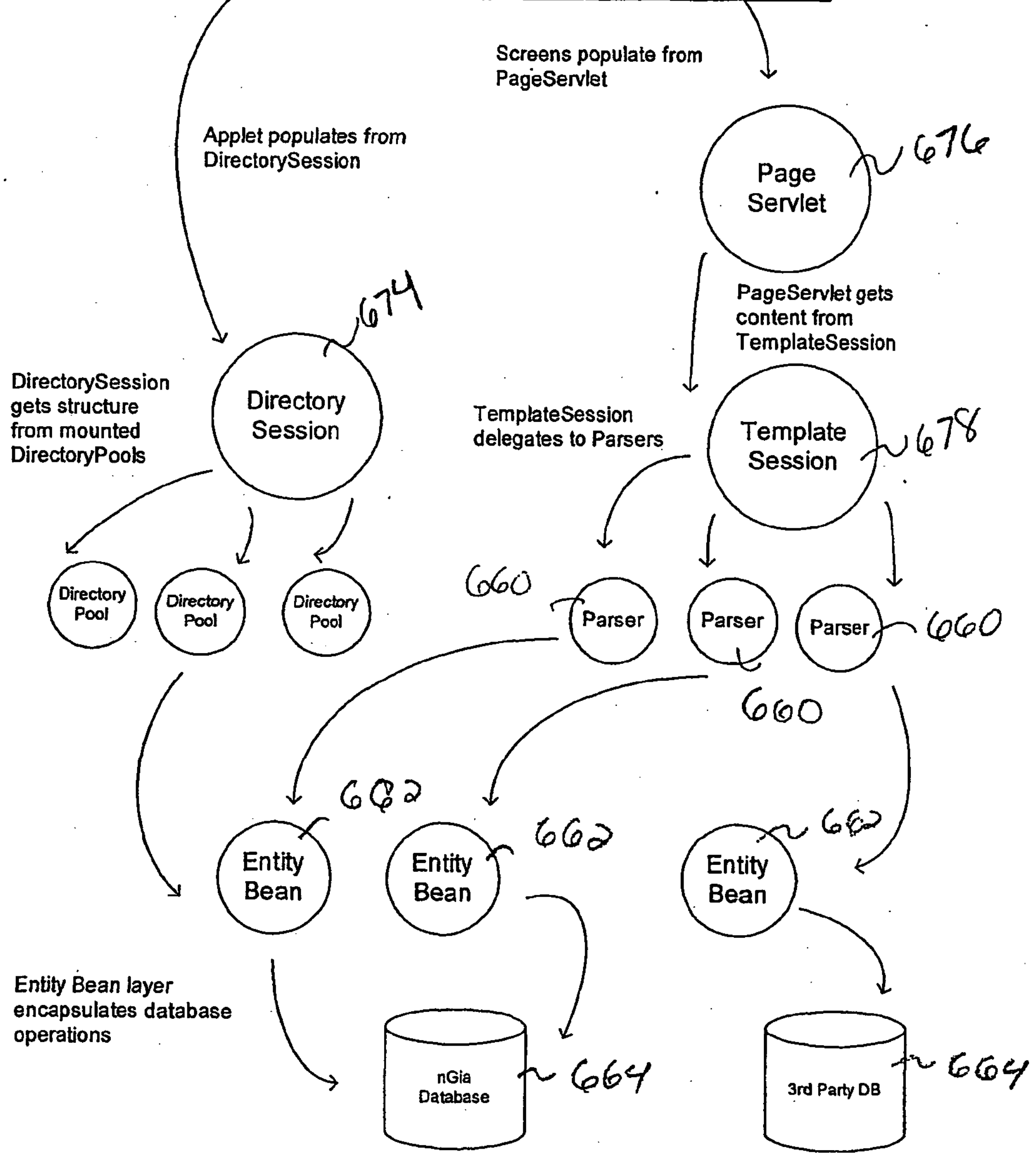

Screens populate from PageServlet
Applet populates from DirectorySession
Page Servlet
676
PageServlet gets content from TemplateSession
674
DirectorySession gets structure from mounted DirectoryPools
Directory Session
TemplateSession delegates to Parsers
Template Session
678
Directory Pool
Directory Pool
Directory Pool
660
Parser
Parser
Parser
660
660
682
662
662
Entity Bean
Entity Bean
Entity Bean
Entity Bean layer encapsulates database operations
nGia Database
664
3rd Party DB
664

Figure 6

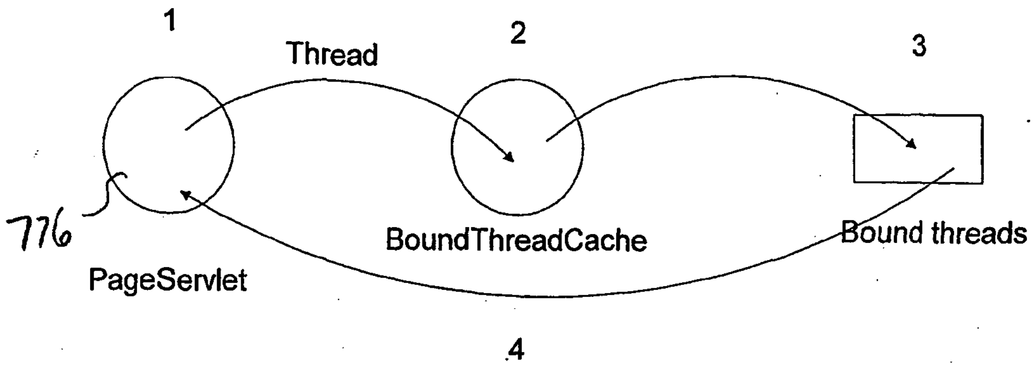
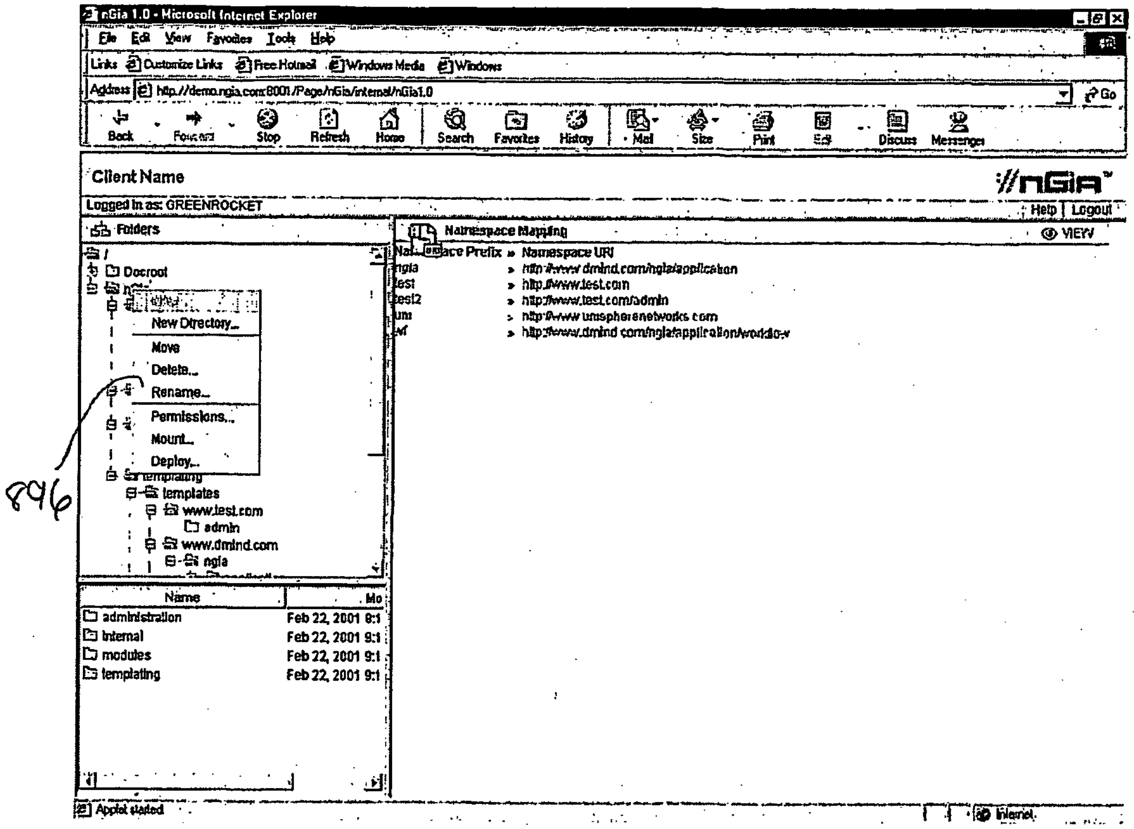

1
Thread
2
3
776
BoundThreadCache
Bound threads
PageServlet
4

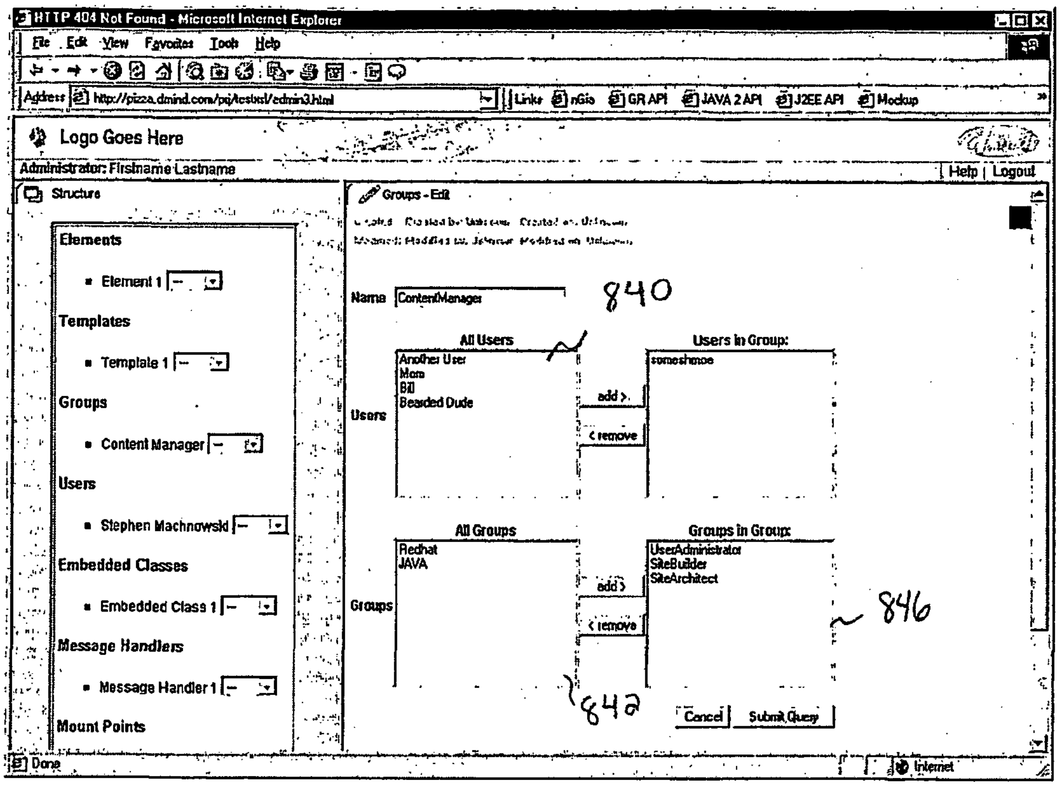
Figure 7


nGia 1.0 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links Customize Links Free Hotmail Windows Media Windows
Address http://demo.ngia.com:8001/Page/nGia/internal/nGia1.0
Go
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss Messenger
Client Name
//nGia™
Logged in as: GREENROCKET
Help | Logout
Folders
/
Docroot
New Directory...
Move
Delete...
Rename...
Permissions...
Mount...
Deploy...
templating
templates
www.test.com
admin
www.dmind.com
ngia
Namespace Mapping
VIEW
Applet started
Internet
896

Namespace Prefix »	Namespace URI
ngia	» http://www.dmind.com/ngia/application
test	» http://www.test.com
test2	» http://www.test.com/admin
uni	» http://www.unispherenetworks.com
wf	» http://www.dmind.com/ngia/application/workflow

Name	Mo
administration	Feb 22, 2001 9:1
internal	Feb 22, 2001 9:1
modules	Feb 22, 2001 9:1
templating	Feb 22, 2001 9:1

Figure 8



HTTP 404 Not Found - Microsoft Internet Explorer

Address: http://pizza.dmind.com/pj/testxsl/admin3.html

Logo Goes Here

Administrator: Firstname Lastname

Structure

Elements

- Element 1

Templates

- Template 1

Groups

- Content Manager

Users

- Stephen Machnowski

Embedded Classes

- Embedded Class 1

Message Handlers

- Message Handler 1

Mount Points

Groups - Edit

Created: Created by: Unknown Created on: Unknown

Modified: Modified by: Unknown Modified on: Unknown

Name: ContentManager

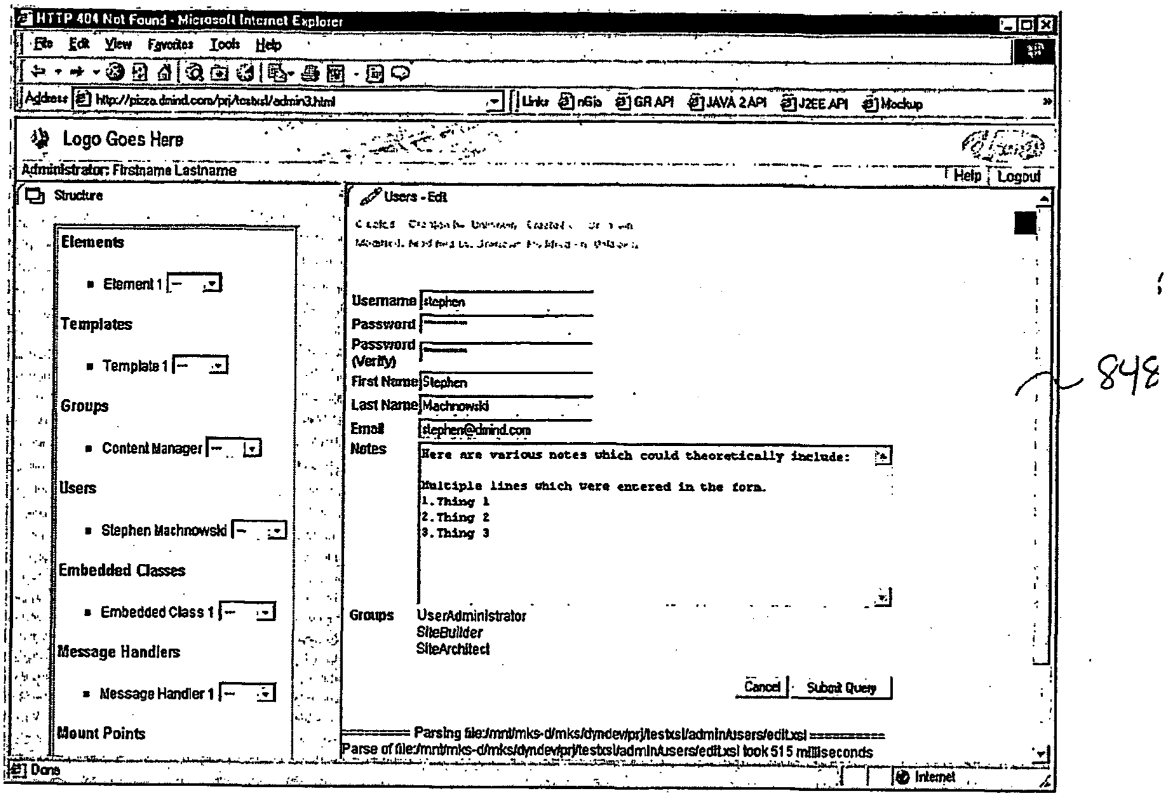
Users

All Users	Users in Group:
Another User Moms Bill Bearded Dude	someshmoe

Groups

All Groups	Groups in Group:
Redhat JAVA	UserAdministrator SiteBuilder SiteArchitect

Figure 8A



[illegible]
[illegible]

Username stephen
Password
Password (Verify)
First Name Stephen
Last Name Machnowski
Email stephen@dnind.com
Notes

```
Here are various notes which could theoretically include:

Multiple lines which were entered in the form.
1.Thing 1
2.Thing 2
3.Thing 3
```

Groups UserAdministrator
SiteBuilder
SiteArchitect

```
======== Parsing file:/mnt/mks-d/mks/dyndev/prj/testxsl/admin/users/edit.xsl ========
Parse of file:/mnt/mks-d/mks/dyndev/prj/testxsl/admin/users/edit.xsl took 515 milliseconds
```

Figure 8B

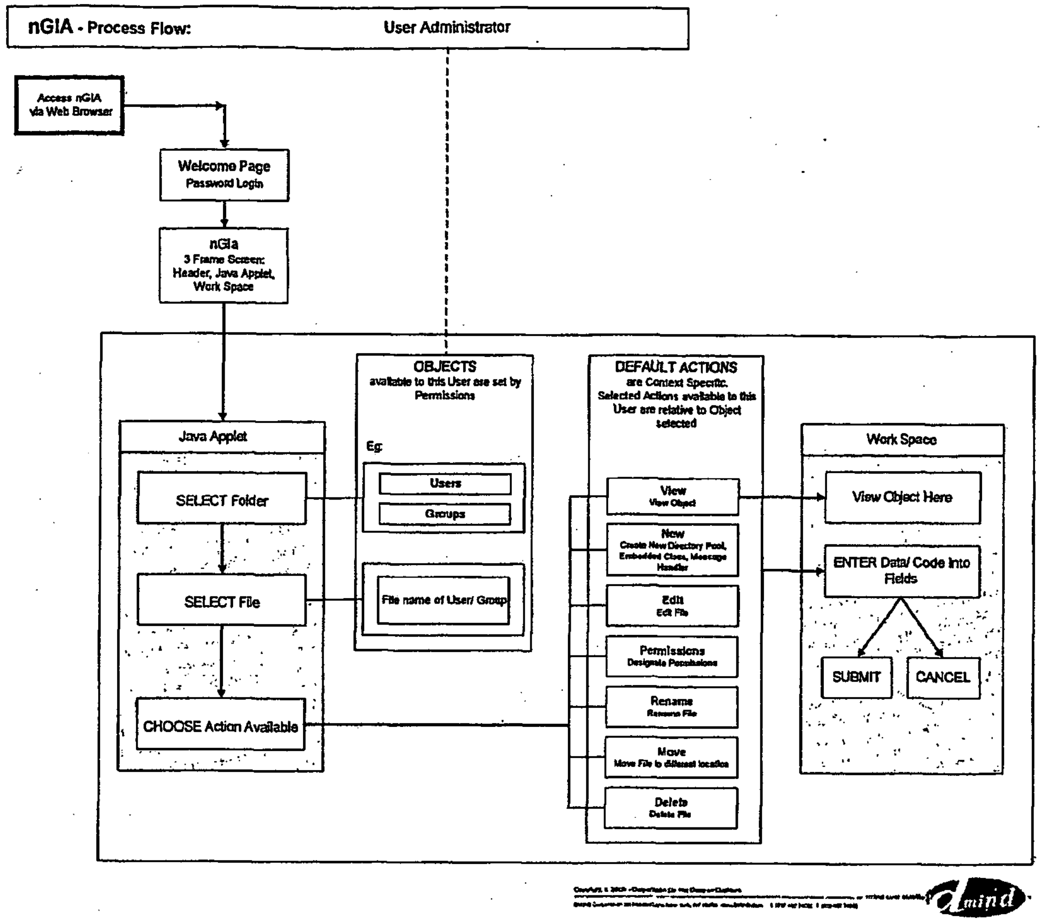
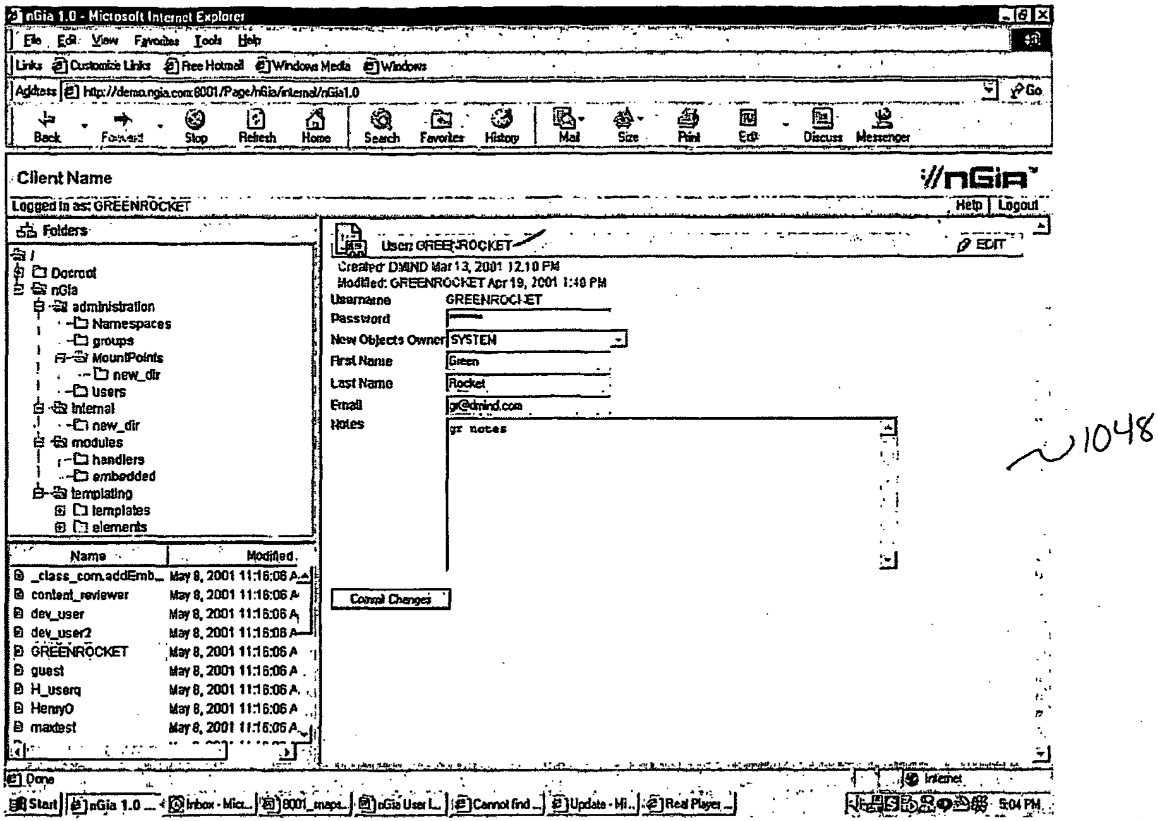

nGIA - Process Flow:
User Administrator
Access nGIA via Web Browser
Welcome Page
Password Login
nGia
3 Frame Screen: Header, Java Applet, Work Space
Java Applet
SELECT Folder
SELECT File
CHOOSE Action Available
OBJECTS
available to this User are set by Permissions
Eg:
Users
Groups
File name of User/ Group
DEFAULT ACTIONS
are Context Specific.
Selected Actions available to this User are relative to Object selected
View
View Object
New
Create New Directory Pool, Embedded Class, Message Handler
Edit
Edit File
Permissions
Designate Permissions
Rename
Rename File
Move
Move File to different location
Delete
Delete File
Work Space
View Object Here
ENTER Data/ Code Into Fields
SUBMIT
CANCEL
mind

Figure 9


nGia 1.0 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links Customize Links Free Hotmail Windows Media Windows
Address http://demo.ngia.com:8001/Page/nGia/internal/nGia1.0
Go
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss Messenger
//nGia
Help Logout
Folders
/
Docroot
nGia
administration
Namespaces
groups
MountPoints
new_dir
users
internal
new_dir
modules
handlers
embedded
templating
templates
elements
EDIT
Cancel Changes
Done
Internet
Start nGia 1.0 ... Inbox - Mic... 8001_snaps... nGia User L... Cannot find ... Update - Mi... Real Player...
5:04 PM
1048

Client Name

Logged in as: GREENROCKET

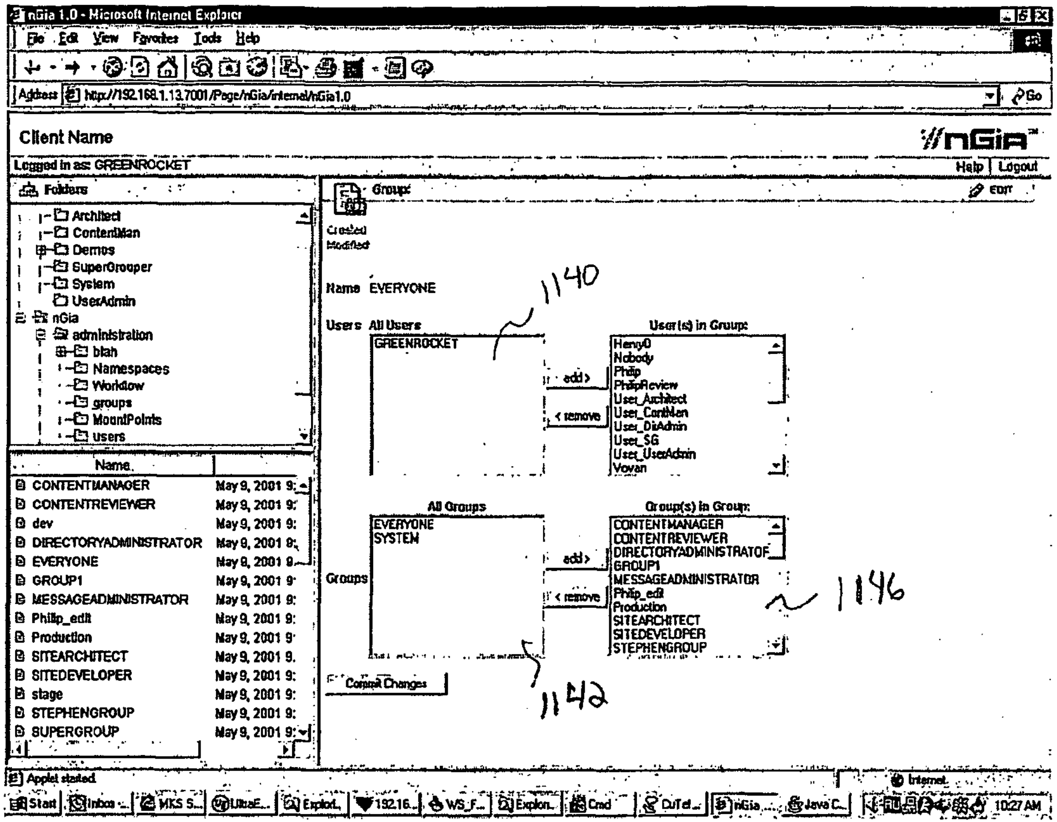
User: GREENROCKET

Created: DMIND Mar 13, 2001 12:10 PM
Modified: GREENROCKET Apr 19, 2001 1:40 PM

Username	GREENROCKET
Password	*******
New Objects Owner	SYSTEM
First Name	Green
Last Name	Rocket
Email	gr@dmind.com
Notes	gr notes

Name	Modified
_class_com.addEmb...	May 8, 2001 11:16:06 A
content_reviewer	May 8, 2001 11:16:06 A
dev_user	May 8, 2001 11:16:06 A
dev_user2	May 8, 2001 11:16:06 A
GREENROCKET	May 8, 2001 11:16:06 A
guest	May 8, 2001 11:16:06 A
H_userq	May 8, 2001 11:16:06 A
HenryO	May 8, 2001 11:16:06 A
maxtest	May 8, 2001 11:16:06 A

Figure 10



Client Name

Logged in as: GREENROCKET

- Architect
- ContentMan
- Demos
- SuperGrouper
- System
- UserAdmin
- nGia
  - administration
    - blah
    - Namespaces
    - Workflow
    - groups
    - MountPoints
    - users

Name.	
CONTENTMANAGER	May 9, 2001 9:
CONTENTREVIEWER	May 9, 2001 9:
dev	May 9, 2001 9:
DIRECTORYADMINISTRATOR	May 9, 2001 9:
EVERYONE	May 9, 2001 9
GROUP1	May 9, 2001 9·
MESSAGEADMINISTRATOR	May 9, 2001 9:
Philip_edit	May 9, 2001 9:
Production	May 9, 2001 9·
SITEARCHITECT	May 9, 2001 9.
SITEDEVELOPER	May 9, 2001 9:
stage	May 9, 2001 9:
STEPHENGROUP	May 9, 2001 9:
SUPERGROUP	May 9, 2001 9:

Created
Modified

Name EVERYONE

Users All Users
GREENROCKET

User(s) in Group:
HenryO
Nobody
Philip
PhilipReview
User_Architect
User_ContMan
User_DirAdmin
User_SG
User_UserAdmin
Vovan

Groups All Groups
EVERYONE
SYSTEM

Group(s) in Group:
CONTENTMANAGER
CONTENTREVIEWER
DIRECTORYADMINISTRATOF
GROUP1
MESSAGEADMINISTRATOR
Philip_edit
Production
SITEARCHITECT
SITEDEVELOPER
STEPHENGROUP

Figure 11

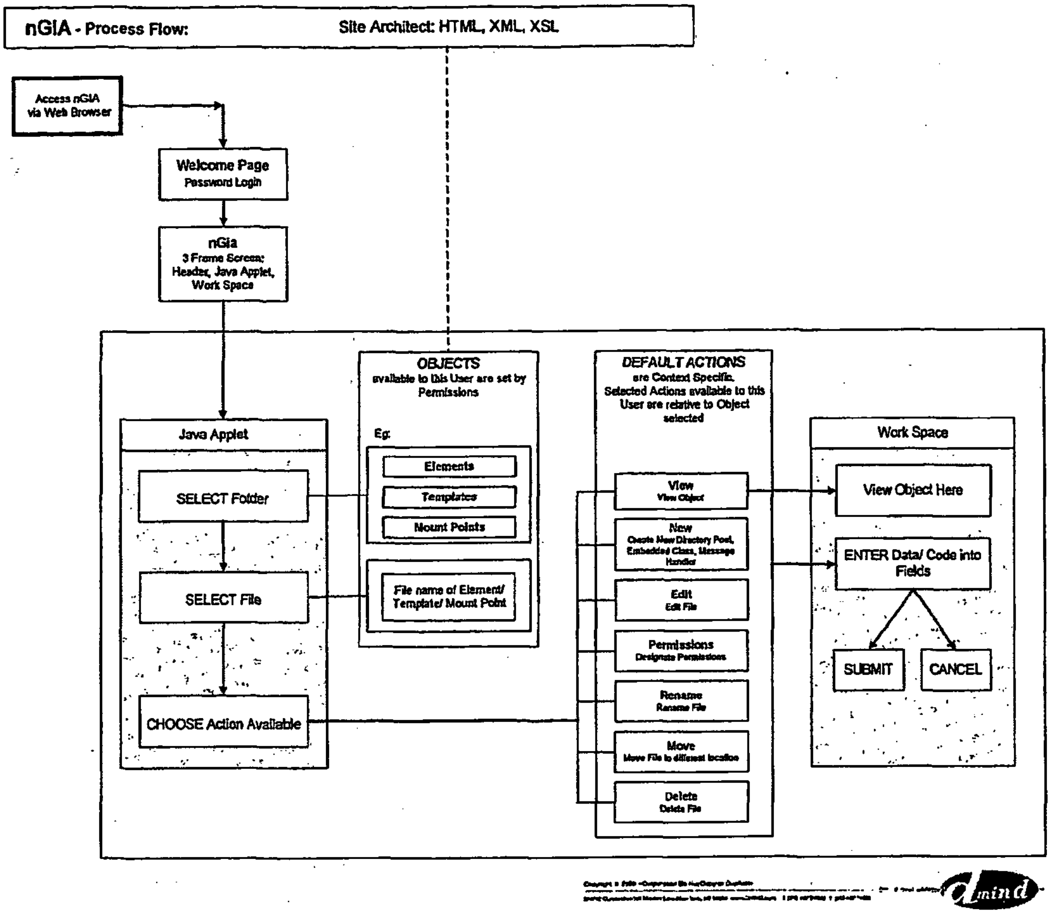
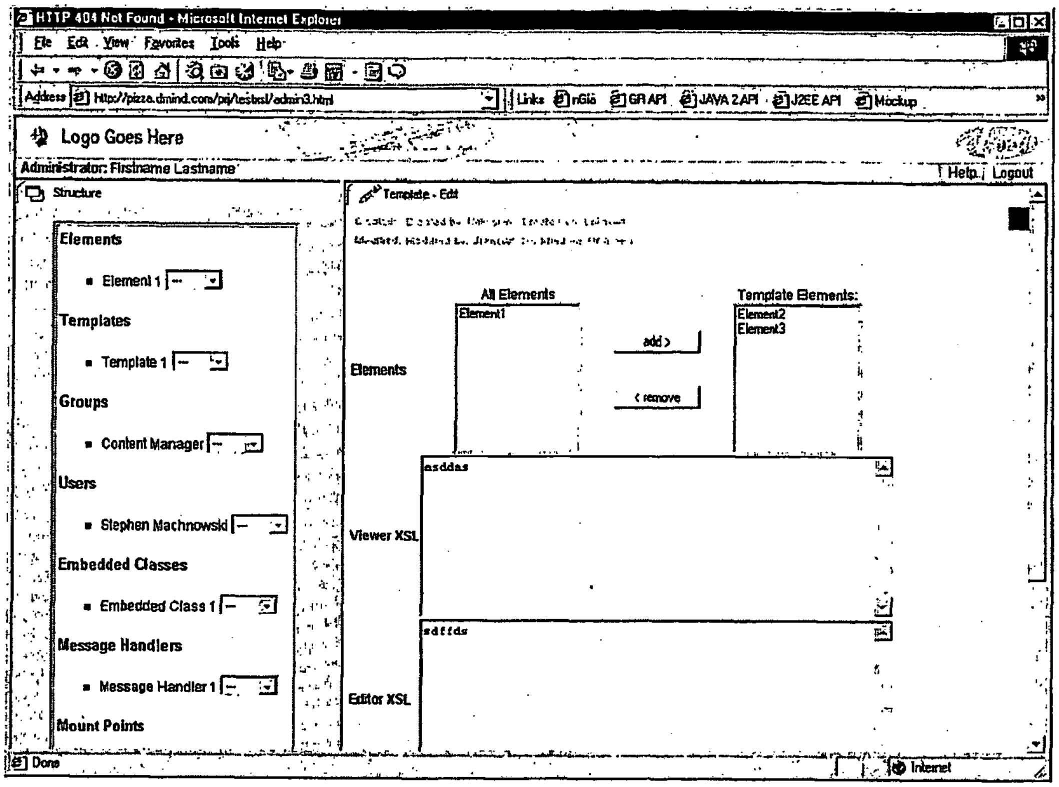

nGIA - Process Flow:
Site Architect: HTML, XML, XSL
Access nGIA via Web Browser
Welcome Page
Password Login
nGia
3 Frame Screen: Header, Java Applet, Work Space
Java Applet
SELECT Folder
SELECT File
CHOOSE Action Available
OBJECTS
available to this User are set by Permissions
Eg:
Elements
Templates
Mount Points
File name of Element/ Template/ Mount Point
DEFAULT ACTIONS
are Context Specific.
Selected Actions available to this User are relative to Object selected
View
View Object
New
Create New Directory Post, Embedded Class, Message Handler
Edit
Edit File
Permissions
Designate Permissions
Rename
Rename File
Move
Move File to different location
Delete
Delete File
Work Space
View Object Here
ENTER Data/ Code into Fields
SUBMIT
CANCEL

Figure 12


HTTP 404 Not Found - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links nGlo GR API JAVA 2 API J2EE API Mockup
Logo Goes Here
Help Logout
Structure
Template - Edit
add >
< remove
Done
Internet

Address http://pizza.dmind.com/pq/testrsl/admin3.html

Administrator: Firstname Lastname

Elements
- Element 1 --
Templates
- Template 1 --
Groups
- Content Manager --
Users
- Stephen Machnowski --
Embedded Classes
- Embedded Class 1 --
Message Handlers
- Message Handler 1 --
Mount Points

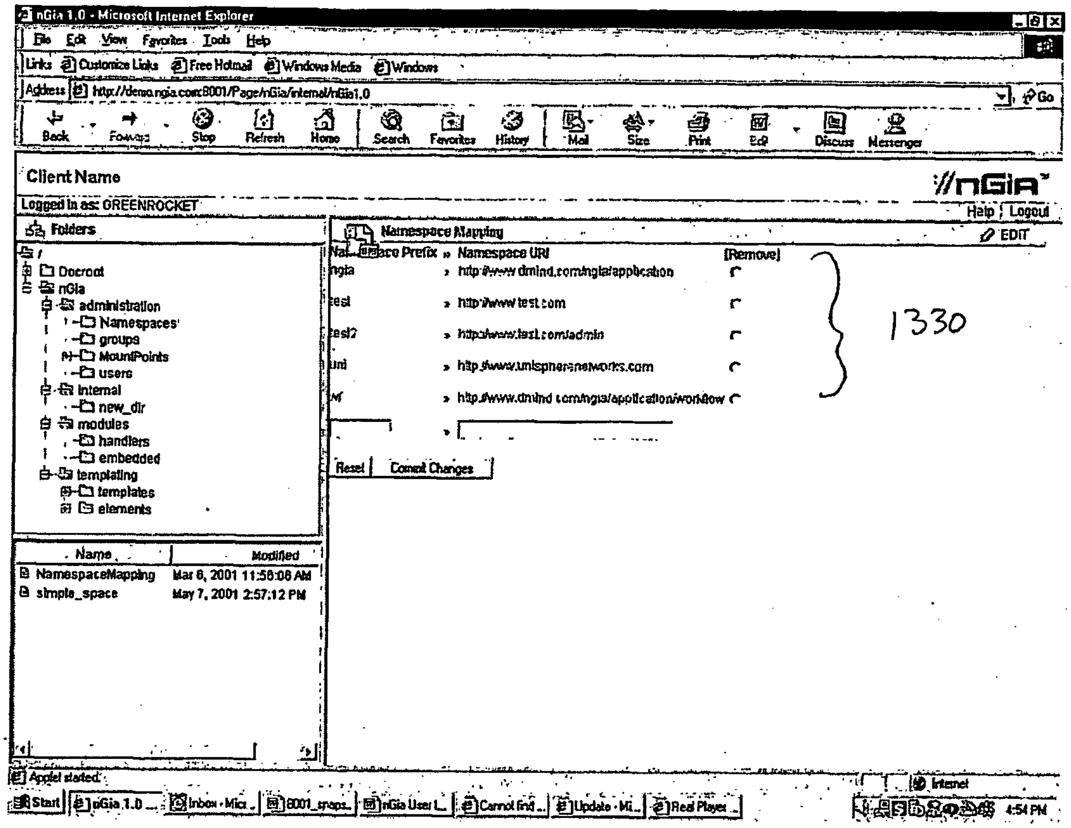
Created: [illegible]
Modified: [illegible]

	All Elements	Template Elements:
Elements	Element1	Element2 Element3

Viewer XSL: asddas

Editor XSL: sdffds

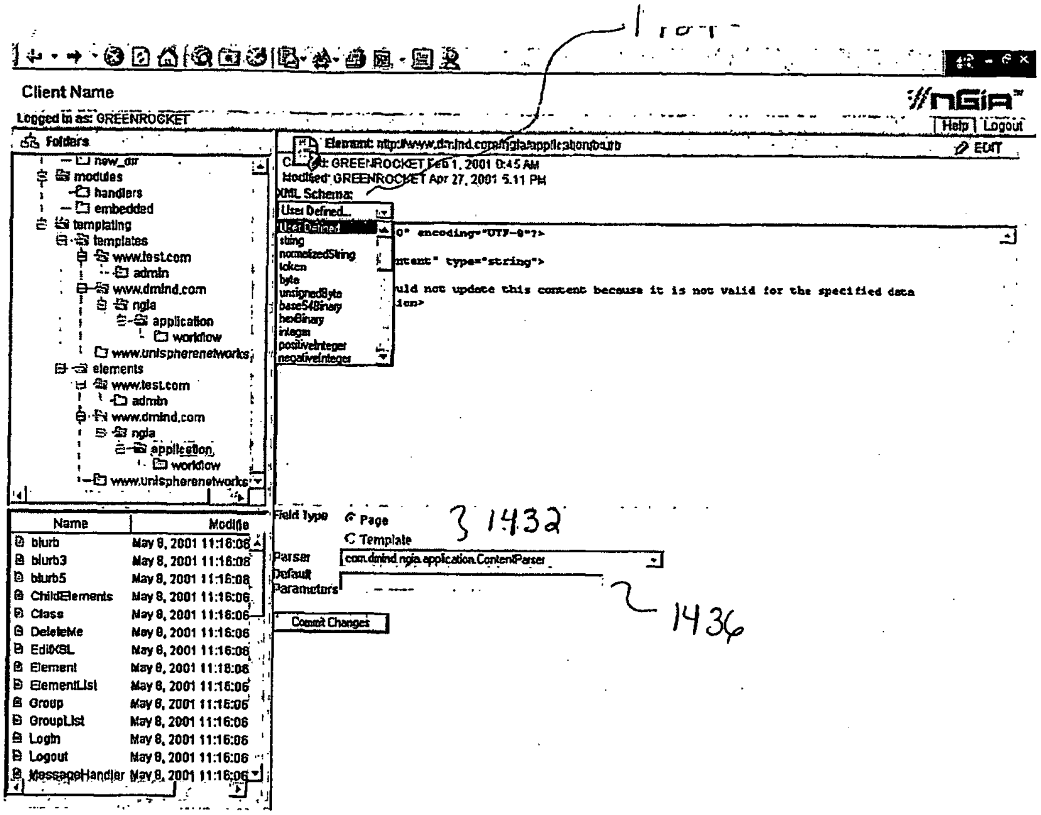
Figure 12A


nGia 1.0 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links Customize Links Free Hotmail Windows Media Windows
Address http://demo.ngia.com:8001/Page/nGia/internal/nGia1.0
Go
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss Messenger
Client Name
//nGia
Logged in as: GREENROCKET
Help Logout
Folders
/
Docroot
nGia
administration
Namespaces
groups
MountPoints
users
internal
new_dir
modules
handlers
embedded
templating
templates
elements
Namespace Mapping
EDIT
Reset
Commit Changes
1330
Applet started
Internet
Start nGia 1.0 ... Inbox - Mic... 8001_snaps... nGia User L... Cannot find ... Update - Mi... Real Player ...
4:54 PM

Namespace Prefix	»	Namespace URI	[Remove]
ngia	»	http://www.dmind.com/ngia/application	○
test	»	http://www.test.com	○
test2	»	http://www.test.com/admin	○
uni	»	http://www.unispherenetworks.com	○
wf	»	http://www.dmind.com/ngia/application/workflow	○
	»		

Name	Modified
NamespaceMapping	Mar 6, 2001 11:56:08 AM
simple_space	May 7, 2001 2:57:12 PM

Figure 13



Logged in as: GREENROCKET

Element: http://www.dmind.com/ngia/application/blurb

Created: GREENROCKET Feb 1, 2001 0:45 AM
Modified: GREENROCKET Apr 27, 2001 5.11 PM

XML Schema:

```
0" encoding="UTF-8"?>

ntent" type="string">

uld not update this content because it is not valid for the specified data
ion>
```

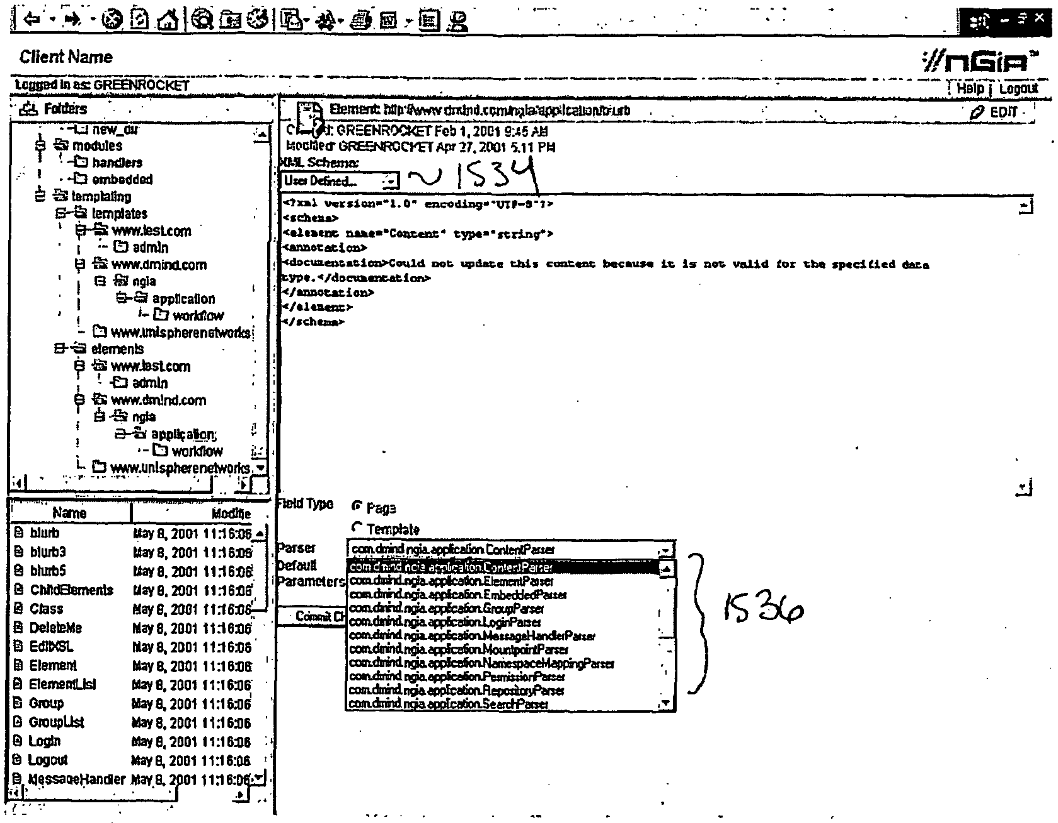
Name	Modifie
blurb	May 8, 2001 11:16:06
blurb3	May 8, 2001 11:16:06
blurb5	May 8, 2001 11:16:06
ChildElements	May 8, 2001 11:16:06
Class	May 8, 2001 11:16:06
DeleteMe	May 8, 2001 11:16:06
EditXSL	May 8, 2001 11:16:06
Element	May 8, 2001 11:16:06
ElementList	May 8, 2001 11:16:06
Group	May 8, 2001 11:16:06
GroupList	May 8, 2001 11:16:06
Login	May 8, 2001 11:16:06
Logout	May 8, 2001 11:16:06
MessageHandler	May 8, 2001 11:16:06

Field Type ● Page ○ Template

Parser com.dmind.ngia.application.ContentParser

Default Parameters

Figure 14



Logged in as: GREENROCKET

Element: http://www.dmind.com/ngia/application/blurb

Created: GREENROCKET Feb 1, 2001 9:45 AM
Modified: GREENROCKET Apr 27, 2001 5:11 PM

XML Schema: User Defined...

```
<?xml version="1.0" encoding="UTF-8"?>
<schema>
<element name="Content" type="string">
<annotation>
<documentation>Could not update this content because it is not valid for the specified data
type.</documentation>
</annotation>
</element>
</schema>
```

Name	Modifie
blurb	May 8, 2001 11:16:06
blurb3	May 8, 2001 11:16:06
blurb5	May 8, 2001 11:16:06
ChildElements	May 8, 2001 11:16:06
Class	May 8, 2001 11:16:06
DeleteMe	May 8, 2001 11:16:06
EditXSL	May 8, 2001 11:16:06
Element	May 8, 2001 11:16:06
ElementList	May 8, 2001 11:16:06
Group	May 8, 2001 11:16:06
GroupList	May 8, 2001 11:16:06
Login	May 8, 2001 11:16:06
Logout	May 8, 2001 11:16:06
MessageHandler	May 8, 2001 11:16:06

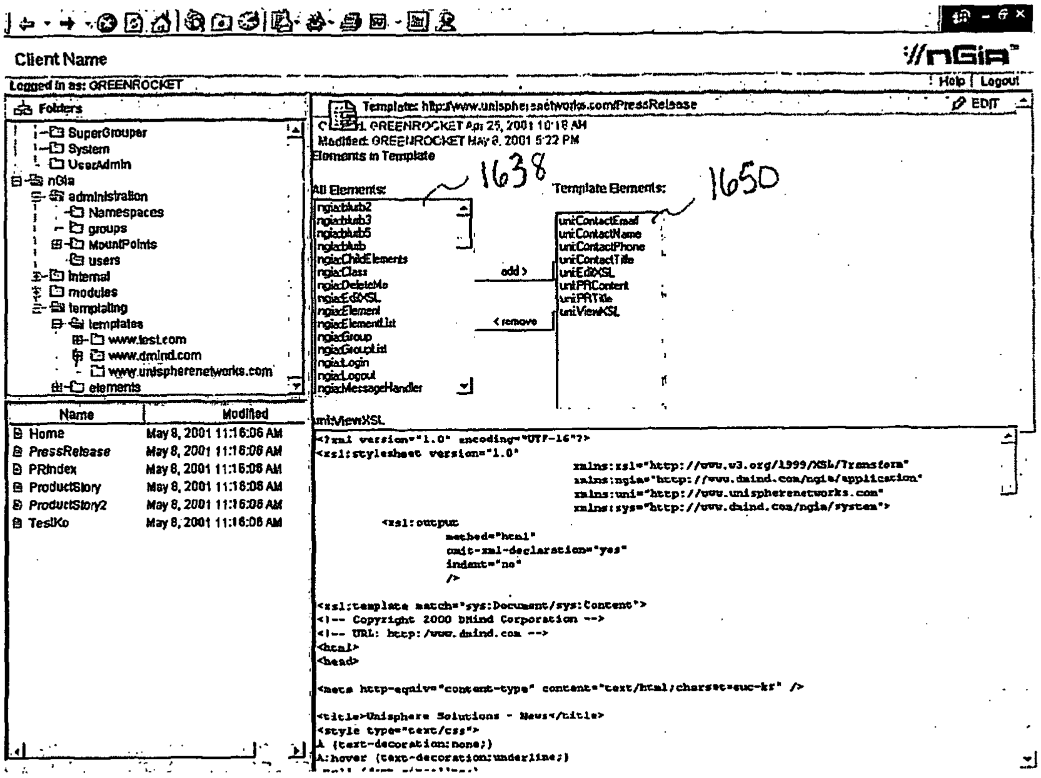
Field Type ● Page ○ Template

Parser com.dmind.ngia.application.ContentParser

Default Parameters

- com.dmind.ngia.application.ContentParser
- com.dmind.ngia.application.ElementParser
- com.dmind.ngia.application.EmbeddedParser
- com.dmind.ngia.application.GroupParser
- com.dmind.ngia.application.LoginParser
- com.dmind.ngia.application.MessageHandlerParser
- com.dmind.ngia.application.MountpointParser
- com.dmind.ngia.application.NamespaceMappingParser
- com.dmind.ngia.application.PermissionParser
- com.dmind.ngia.application.RepositoryParser
- com.dmind.ngia.application.SearchParser

Figure 15



Logged in as: GREENROCKET

Name	Modified
Home	May 8, 2001 11:16:06 AM
PressRelease	May 8, 2001 11:16:06 AM
PRIndex	May 8, 2001 11:16:06 AM
ProductStory	May 8, 2001 11:16:06 AM
ProductStory2	May 8, 2001 11:16:06 AM
TestKo	May 8, 2001 11:16:06 AM

Template: http://www.unispherenetworks.com/PressRelease

Created: GREENROCKET Apr 25, 2001 10:18 AM

Modified: GREENROCKET May 8, 2001 5:22 PM

Elements in Template

All Elements:

ngia:blurb2
ngia:blurb3
ngia:blurb5
ngia:blurb
ngia:ChildElements
ngia:Class
ngia:DeleteMe
ngia:EditXSL
ngia:Element
ngia:ElementList
ngia:Group
ngia:GroupList
ngia:Login
ngia:Logout
ngia:MessageHandler

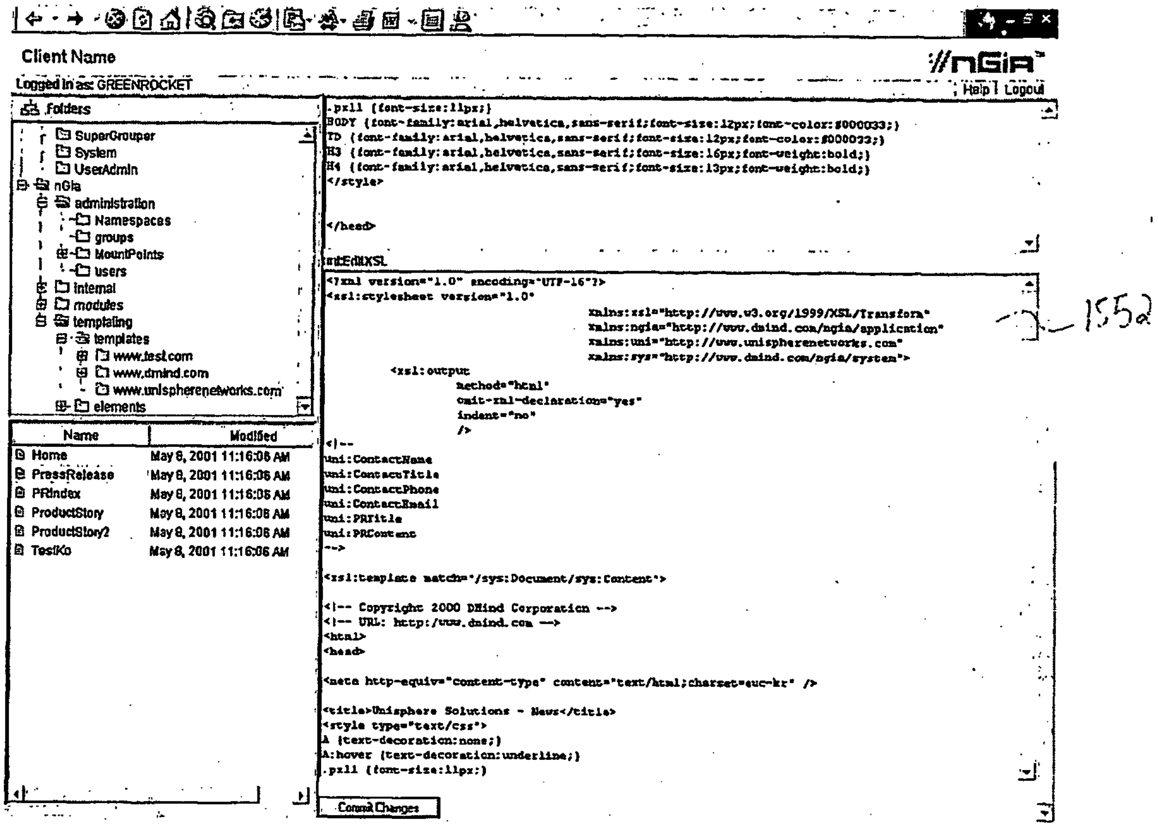
Template Elements:

uni:ContactEmail
uni:ContactName
uni:ContactPhone
uni:ContactTitle
uni:EditXSL
uni:PRContent
uni:PRTitle
uni:ViewXSL

uni:ViewXSL

```
<?xml version="1.0" encoding="UTF-16"?>
<xsl:stylesheet version="1.0"
                                        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                        xmlns:ngia="http://www.dmind.com/ngia/application"
                                        xmlns:uni="http://www.unispherenetworks.com"
                                        xmlns:sys="http://www.dmind.com/ngia/system">

         <xsl:output
                  method="html"
                  omit-xml-declaration="yes"
                  indent="no"
                  />

<xsl:template match="sys:Document/sys:Content">
<!-- Copyright 2000 DMind Corporation -->
<!-- URL: http:/www.dmind.com -->
<html>
<head>

<meta http-equiv="content-type" content="text/html;charset=euc-kr" />

<title>Unisphere Solutions - News</title>
<style type="text/css">
A {text-decoration:none;}
A:hover {text-decoration:underline;}
```

Figure 16



Name	Modified
Home	May 8, 2001 11:16:06 AM
PressRelease	May 8, 2001 11:16:06 AM
PRIndex	May 8, 2001 11:16:06 AM
ProductStory	May 8, 2001 11:16:06 AM
ProductStory2	May 8, 2001 11:16:06 AM
TestKo	May 8, 2001 11:16:06 AM

```
.pxll {font-size:11px;}
BODY {font-family:arial,helvetica,sans-serif;font-size:12px;font-color:#000033;}
TD {font-family:arial,helvetica,sans-serif;font-size:12px;font-color:#000033;}
H3 {font-family:arial,helvetica,sans-serif;font-size:16px;font-weight:bold;}
H4 {font-family:arial,helvetica,sans-serif;font-size:13px;font-weight:bold;}
</style>


</head>
```

```
<?xml version="1.0" encoding="UTF-16"?>
<xsl:stylesheet version="1.0"
                                        xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                        xmlns:ngia="http://www.dmind.com/ngia/application"
                                        xmlns:uni="http://www.unispherenetworks.com"
                                        xmlns:sys="http://www.dmind.com/ngia/system">
        <xsl:output
                method="html"
                omit-xml-declaration="yes"
                indent="no"
                />
<!--
uni:ContactName
uni:ContactTitle
uni:ContactPhone
uni:ContactEmail
uni:PRTitle
uni:PRContent
-->

<xsl:template match="/sys:Document/sys:Content">

<!-- Copyright 2000 DMind Corporation -->
<!-- URL: http:/www.dmind.com -->
<html>
<head>

<meta http-equiv="content-type" content="text/html;charset=euc-kr" />

<title>Unisphere Solutions - News</title>
<style type="text/css">
A {text-decoration:none;}
A:hover {text-decoration:underline;}
.pxll {font-size:11px;}
```

Figure 17

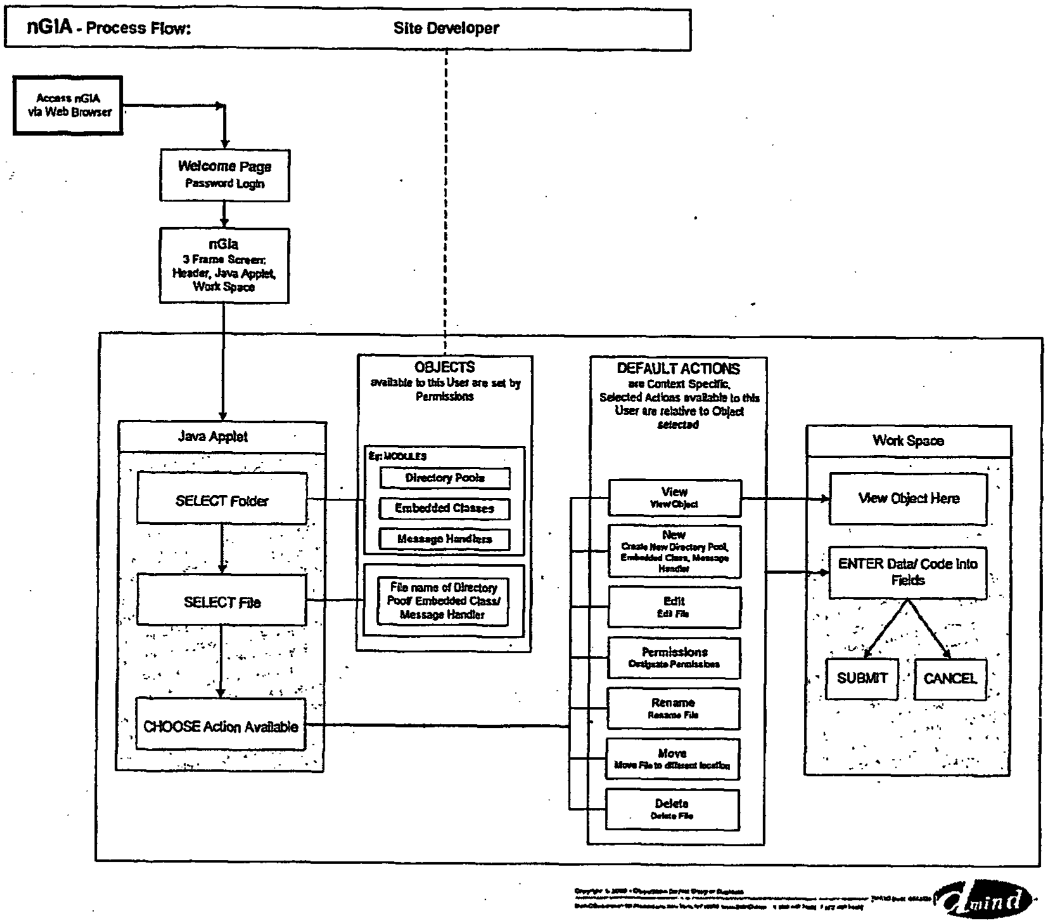
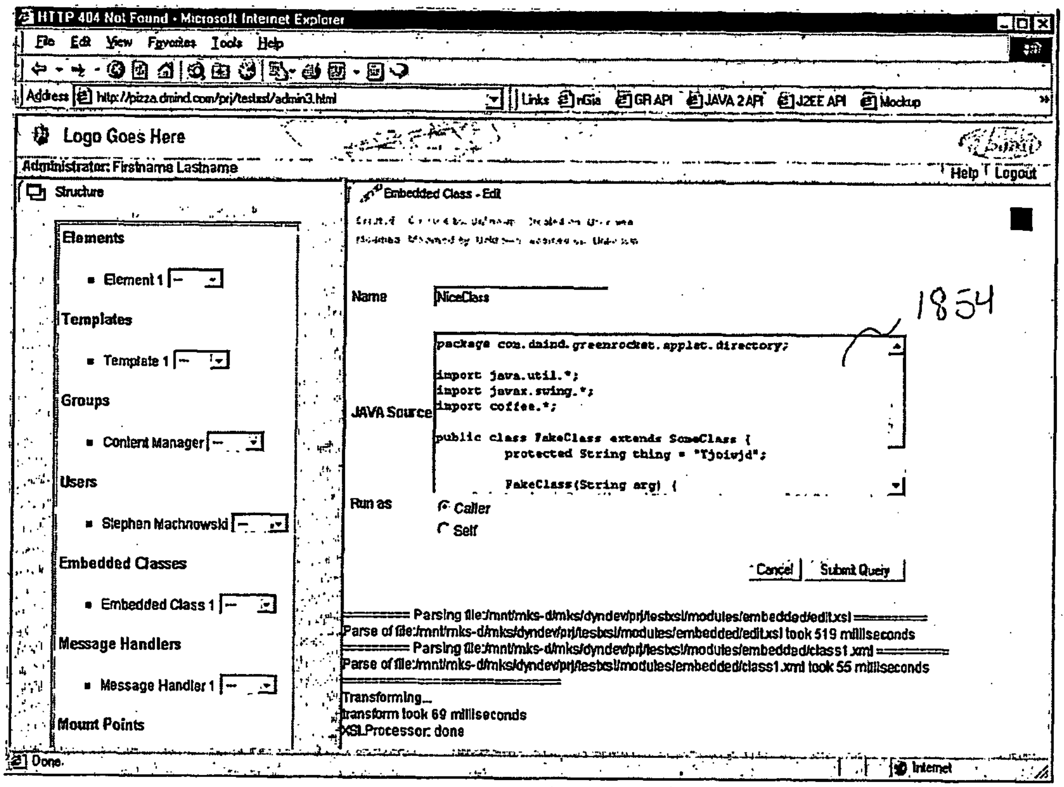

nGIA - Process Flow:
Site Developer
Access nGIA via Web Browser
Welcome Page
Password Login
nGia
3 Frame Screen: Header, Java Applet, Work Space
Java Applet
SELECT Folder
SELECT File
CHOOSE Action Available
OBJECTS
available to this User are set by Permissions
Eg: MODULES
Directory Pools
Embedded Classes
Message Handlers
File name of Directory Pool/ Embedded Class/ Message Handler
DEFAULT ACTIONS
are Context Specific, Selected Actions available to this User are relative to Object selected
View
View Object
New
Create New Directory Pool, Embedded Class, Message Handler
Edit
Edit File
Permissions
Designate Permissions
Rename
Rename File
Move
Move File to different location
Delete
Delete File
Work Space
View Object Here
ENTER Data/ Code Into Fields
SUBMIT
CANCEL

Figure 18



Administrator: Firstname Lastname

Name NiceClass

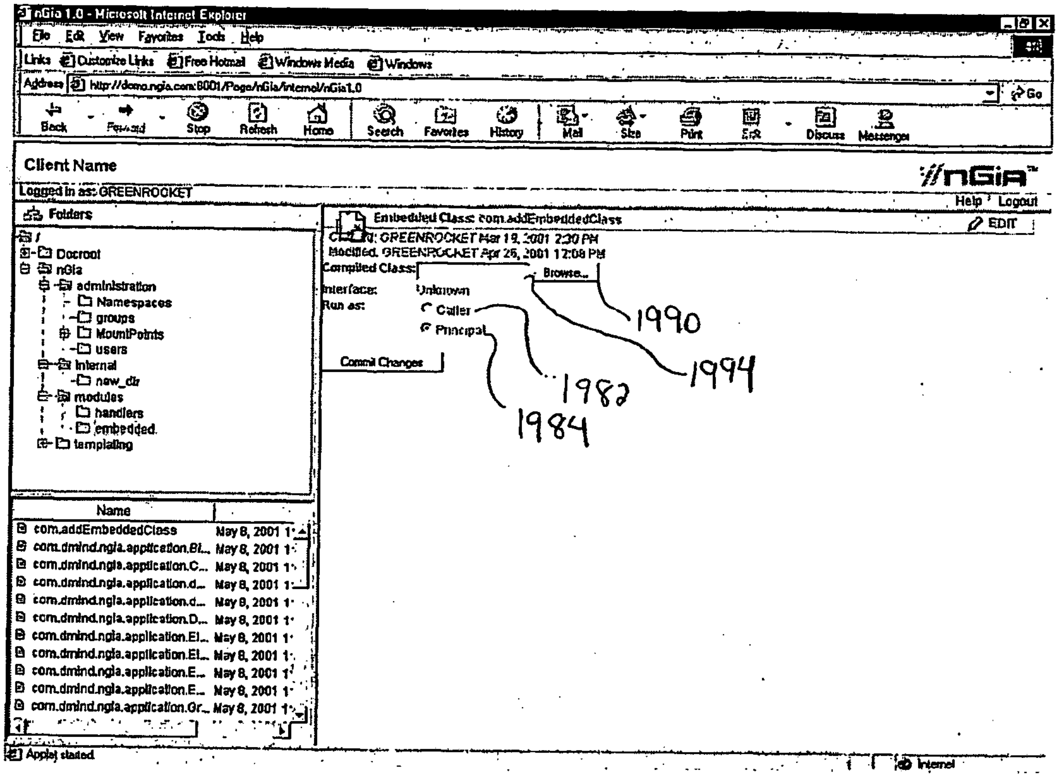
JAVA Source

```
package com.dmind.greenrocket.applet.directory;

import java.util.*;
import javax.swing.*;
import coffee.*;

public class FakeClass extends SomeClass {
        protected String thing = "Tjoiwjd";

        FakeClass(String arg) {
```

Run as ● Caller ○ Self

```
======== Parsing file:/mnt/mks-d/mks/dyndev/prj/testxsl/modules/embedded/edit.xsl ========
Parse of file:/mnt/mks-d/mks/dyndev/prj/testxsl/modules/embedded/edit.xsl took 519 milliseconds
======== Parsing file:/mnt/mks-d/mks/dyndev/prj/testxsl/modules/embedded/class1.xml ========
Parse of file:/mnt/mks-d/mks/dyndev/prj/testxsl/modules/embedded/class1.xml took 55 milliseconds
========================
Transforming...
transform took 69 milliseconds
XSLProcessor: done
```

Figure 18A


nGia 1.0 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links Customize Links Free Hotmail Windows Media Windows
Address http://demo.ngia.com:8001/Page/nGia/internal/nGia1.0
Go
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss Messenger
//nGia™
Help Logout
EDIT
Folders
Browse...
Commit Changes
Applet started
Internet
1990
1994
1982
1984

Client Name

Logged in as: GREENROCKET

- /
  - Docroot
  - nGia
    - administration
      - Namespaces
      - groups
      - MountPoints
      - users
    - internal
      - new_db
    - modules
      - handlers
      - embedded
    - templating

Name	
com.addEmbeddedClass	May 8, 2001 1
com.dmind.ngia.application.Bl...	May 8, 2001 1
com.dmind.ngia.application.C...	May 8, 2001 1
com.dmind.ngia.application.d...	May 8, 2001 1
com.dmind.ngia.application.d...	May 8, 2001 1
com.dmind.ngia.application.D...	May 8, 2001 1
com.dmind.ngia.application.El...	May 8, 2001 1
com.dmind.ngia.application.El...	May 8, 2001 1
com.dmind.ngia.application.E...	May 8, 2001 1
com.dmind.ngia.application.E...	May 8, 2001 1
com.dmind.ngia.application.Gr...	May 8, 2001 1

Embedded Class: com.addEmbeddedClass

Created: GREENROCKET Mar 19, 2001 2:30 PM

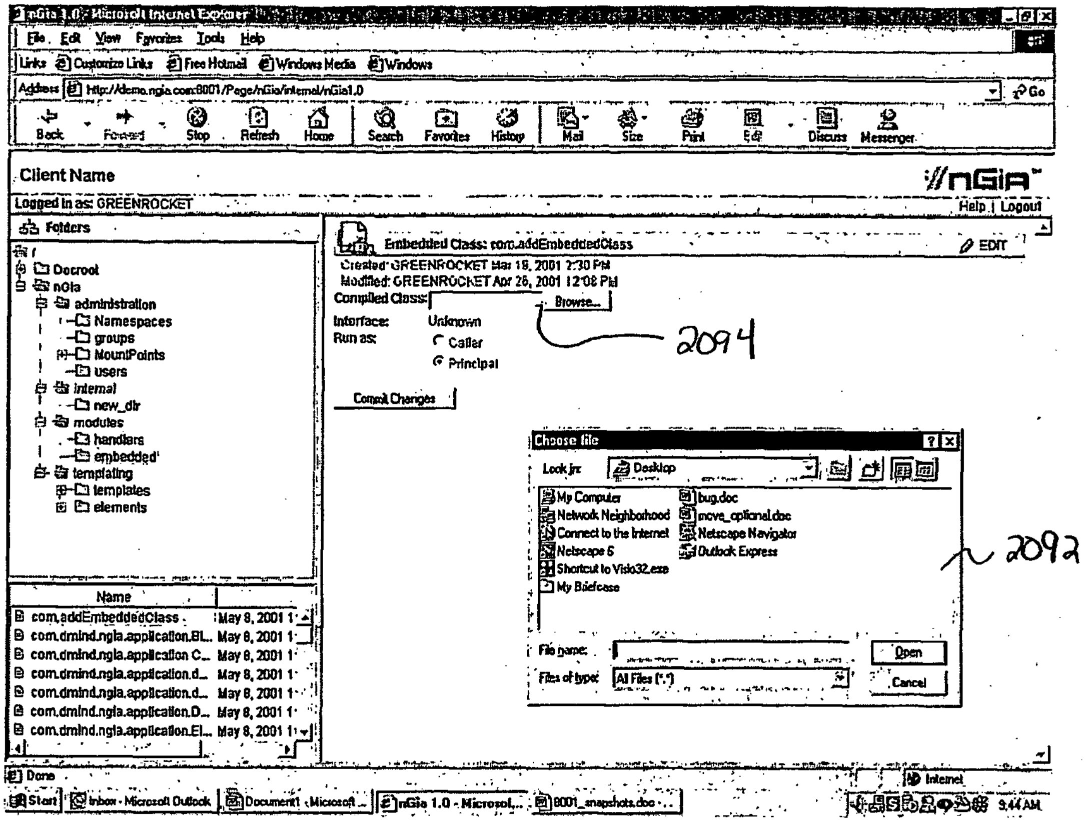
Modified: GREENROCKET Apr 25, 2001 12:08 PM

Compiled Class:

Interface: Unknown

Run as: ○ Caller ● Principal

Figure 19


nGia 1.0 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links Customize Links Free Hotmail Windows Media Windows
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss Messenger
Go
Client Name
//nGiA
Help | Logout
Folders
EDIT
Browse...
Commit Changes
Choose file
Open
Cancel
Name
Done
Internet
Start
Inbox - Microsoft Outlook
Document1 - Microsoft ...
nGia 1.0 - Microsof...
8001_snapshots.doc - ...
9:44 AM
2094
2092

Address http://demo.ngia.com:8001/Page/nGia/internal/nGia1.0

Logged in as: GREENROCKET

```
/
  Docroot
  nGia
    administration
      Namespaces
      groups
      MountPoints
      users
    internal
      new_dir
    modules
      handlers
      embedded
    templating
      templates
      elements
```

Name	
com.addEmbeddedClass	May 8, 2001 1
com.dmind.ngia.application.BL...	May 8, 2001 1
com.dmind.ngia.application.C...	May 8, 2001 1
com.dmind.ngia.application.d...	May 8, 2001 1
com.dmind.ngia.application.d...	May 8, 2001 1
com.dmind.ngia.application.D...	May 8, 2001 1
com.dmind.ngia.application.El...	May 8, 2001 1

Embedded Class: com.addEmbeddedClass

Created: GREENROCKET Mar 18, 2001 2:30 PM

Modified: GREENROCKET Apr 26, 2001 12:08 PM

Compiled Class:

Interface: Unknown

Run as: ○ Caller ● Principal

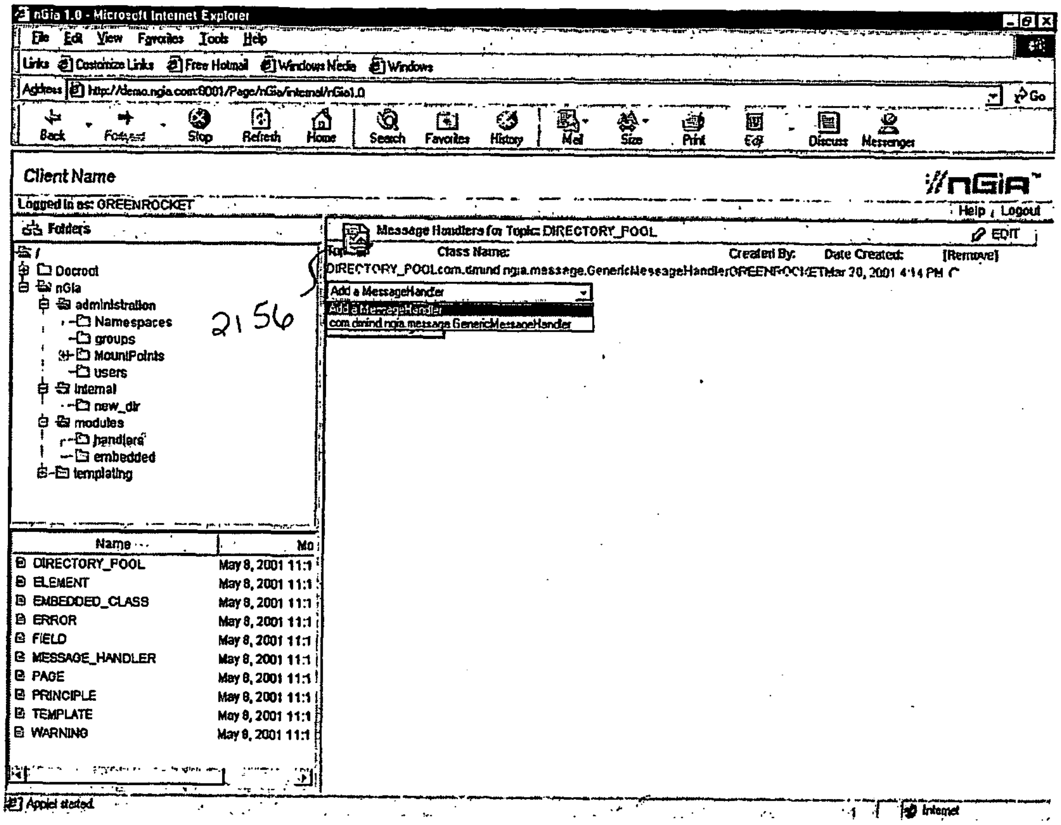
Look in: Desktop

My Computer
Network Neighborhood
Connect to the Internet
Netscape 6
Shortcut to Visio32.exe
My Briefcase
bug.doc
move_optional.doc
Netscape Navigator
Outlook Express

File name:

Files of type: All Files (*.*)

Figure 20


nGia 1.0 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links Customize Links Free Hotmail Windows Media Windows
Address http://demo.ngia.com:8001/Page/nGia/internal/nGia1.0
Go
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss Messenger
//nGIA™
Help | Logout
Folders
EDIT
2156
Applet started
Internet

Client Name

Logged in as: GREENROCKET

- /
  - Docroot
  - nGia
    - administration
      - Namespaces
      - groups
      - MountPoints
      - users
    - internal
      - new_dir
    - modules
      - handlers
      - embedded
    - templating

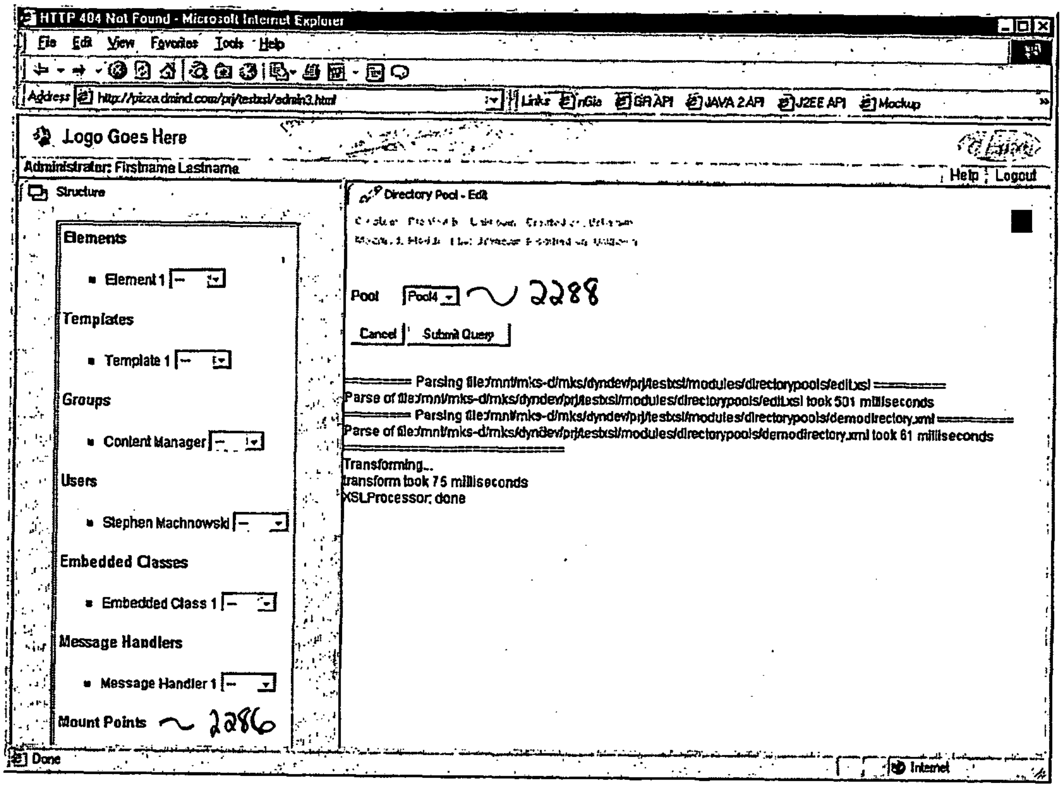
Name	Mo
DIRECTORY_POOL	May 8, 2001 11:1
ELEMENT	May 8, 2001 11:1
EMBEDDED_CLASS	May 8, 2001 11:1
ERROR	May 8, 2001 11:1
FIELD	May 8, 2001 11:1
MESSAGE_HANDLER	May 8, 2001 11:1
PAGE	May 8, 2001 11:1
PRINCIPLE	May 8, 2001 11:1
TEMPLATE	May 8, 2001 11:1
WARNING	May 8, 2001 11:1

Message Handlers for Topic: DIRECTORY_POOL

Topic	Class Name:	Created By:	Date Created:	[Remove]
DIRECTORY_POOL	com.dmind.ngia.message.GenericMessageHandler	GREENROCKET	Mar 20, 2001 4:14 PM	○

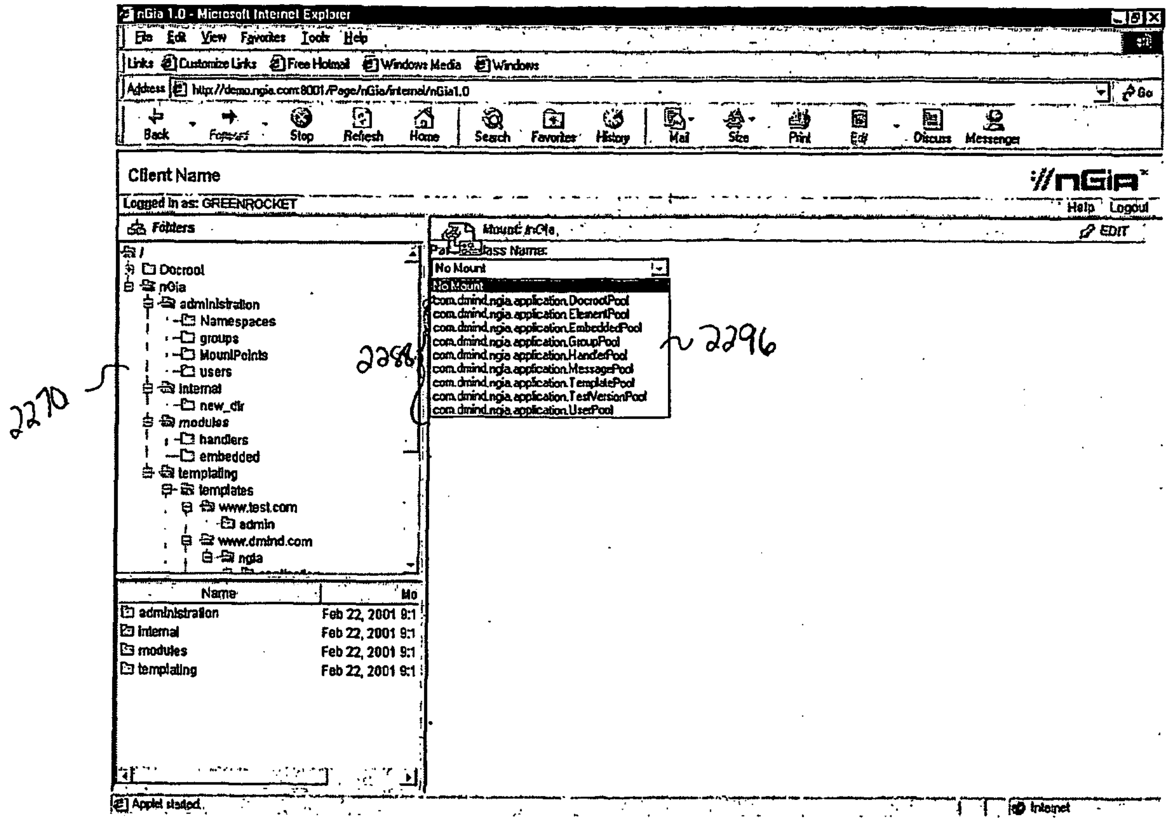
Add a MessageHandler
- Add a MessageHandler
- com.dmind.ngia.message.GenericMessageHandler

Figure 21



```
======== Parsing file:/mnt/mks-d/mks/dyndev/prj/testxsl/modules/directorypools/edit.xsl ========
Parse of file:/mnt/mks-d/mks/dyndev/prj/testxsl/modules/directorypools/edit.xsl took 501 milliseconds
======== Parsing file:/mnt/mks-d/mks/dyndev/prj/testxsl/modules/directorypools/demodirectory.xml ==========
Parse of file:/mnt/mks-d/mks/dyndev/prj/testxsl/modules/directorypools/demodirectory.xml took 61 milliseconds
==========================
Transforming...
transform took 75 milliseconds
XSLProcessor: done
```

Figure 22


nGia 1.0 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links Customize Links Free Hotmail Windows Media Windows
Address
Go
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss Messenger
//nGia
Help Logout
Folders
EDIT
Name
Mo
Internet
2270
2288
2296

Address: http://demo.ngia.com:8001/Page/nGia/internal/nGia1.0

Client Name

Logged in as: GREENROCKET

/
Docroot
nGia
administration
Namespaces
groups
MountPoints
users
internal
new_dir
modules
handlers
embedded
templating
templates
www.test.com
admin
www.dmind.com
ngia

Mount nGia,

Pa[illegible]lass Name:

No Mount

No Mount
com.dmind.ngia.application.DocrootPool
com.dmind.ngia.application.ElementPool
com.dmind.ngia.application.EmbeddedPool
com.dmind.ngia.application.GroupPool
com.dmind.ngia.application.HandlerPool
com.dmind.ngia.application.MessagePool
com.dmind.ngia.application.TemplatePool
com.dmind.ngia.application.TestVersionPool
com.dmind.ngia.application.UserPool

Name	Mo
administration	Feb 22, 2001 9:1
internal	Feb 22, 2001 9:1
modules	Feb 22, 2001 9:1
templating	Feb 22, 2001 9:1

Applet started.

Figure 22A

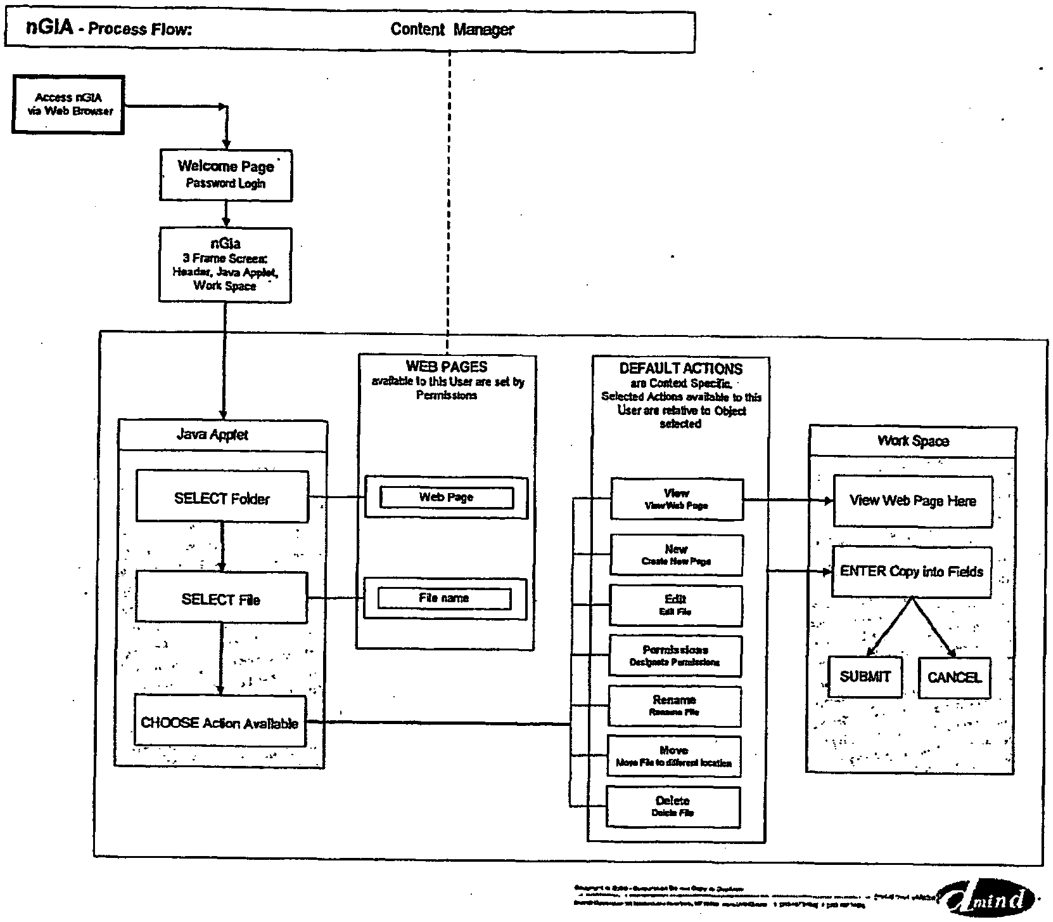
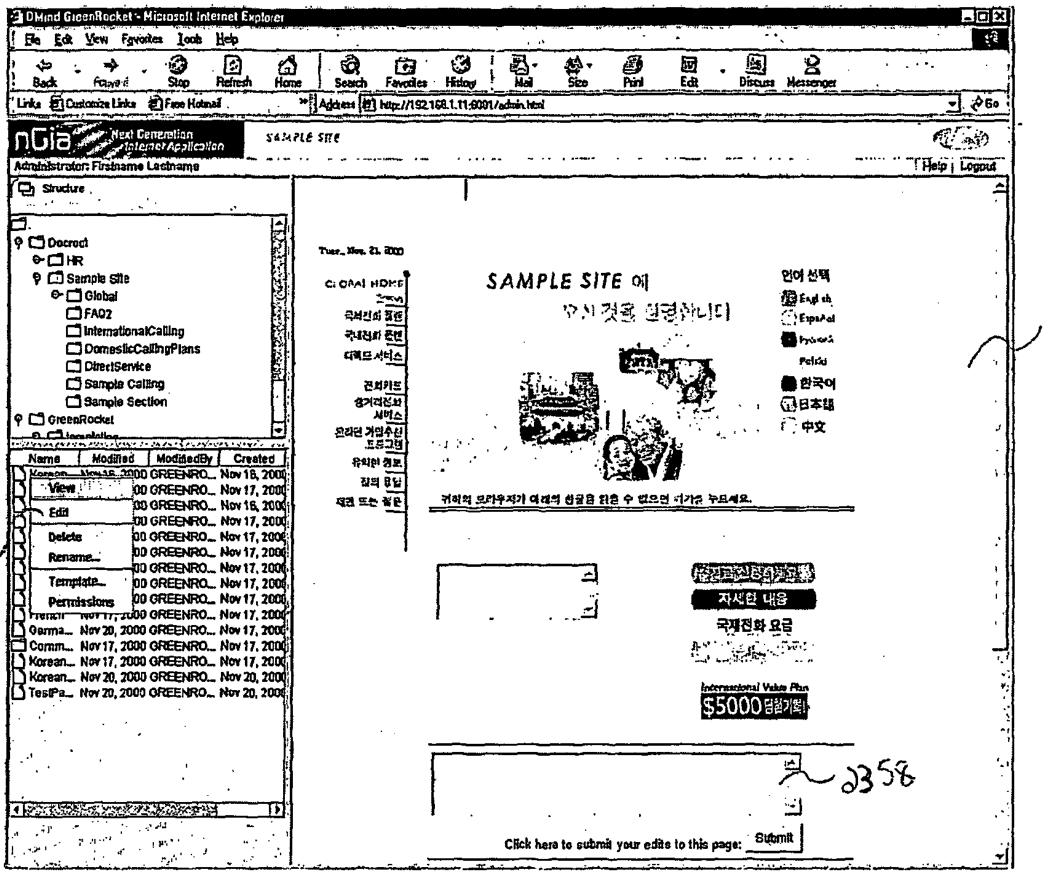
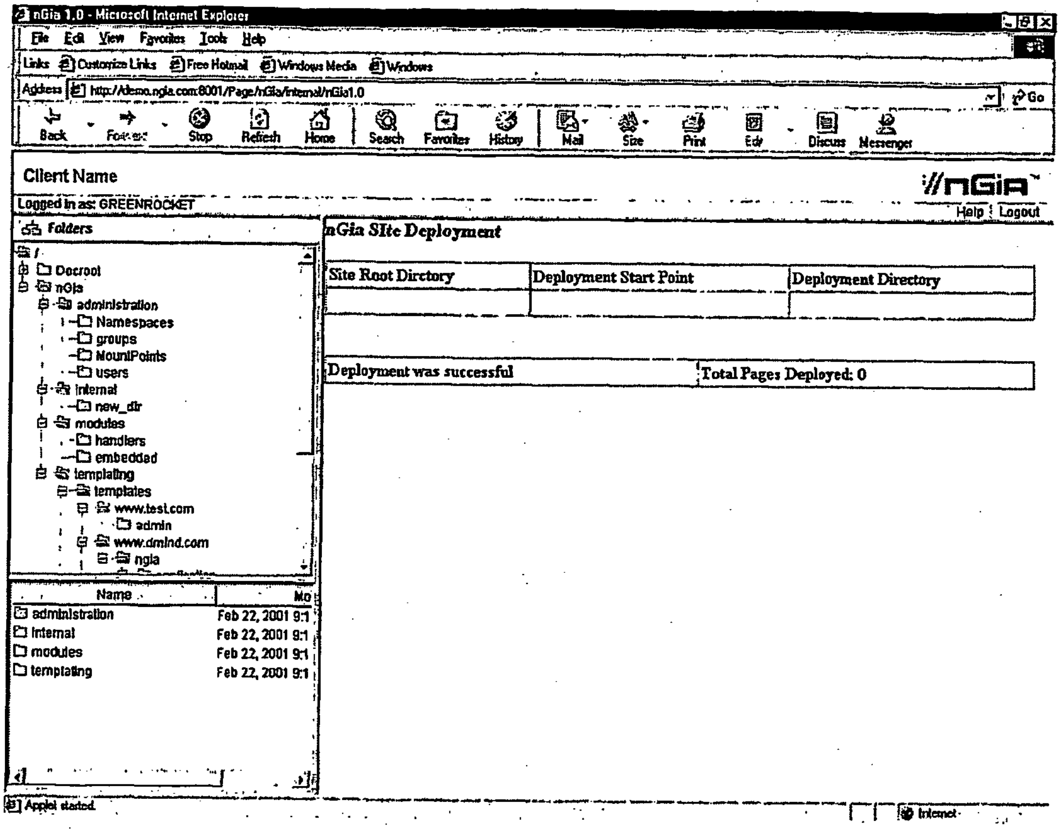

nGIA - Process Flow:
Content Manager
Access nGIA via Web Browser
Welcome Page
Password Login
nGIa
3 Frame Screen: Header, Java Applet, Work Space
Java Applet
SELECT Folder
SELECT File
CHOOSE Action Available
WEB PAGES
available to this User are set by Permissions
Web Page
File name
DEFAULT ACTIONS
are Context Specific.
Selected Actions available to this User are relative to Object selected
View
View Web Page
New
Create New Page
Edit
Edit File
Permissions
Designate Permissions
Rename
Rename File
Move
Move File to different location
Delete
Delete File
Work Space
View Web Page Here
ENTER Copy into Fields
SUBMIT
CANCEL
mind

Figure 23


DMind GreenRocket - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss Messenger
Links Customize Links Free Hotmail Address http://192.168.1.11:8001/admin.html Go
nGia Next Generation Internet Application
SAMPLE SITE
Administrator: Firstname Lastname
Help | Logout
Structure
Docroot
HR
Sample Site
Global
FAQ2
InternationalCalling
DomesticCallingPlans
DirectService
Sample Calling
Sample Section
GreenRocket
View
Edit
Delete
Rename...
Template...
Permissions
Tue., Nov. 21, 2000
GLOBAL HOME
SAMPLE SITE 에
언어 선택
English
Español
Polski
한국어
日本語
中文
자세한 내용
국제전화 요금
International Value Plan
$5000
Click here to submit your edits to this page:
Submit
2372
2396
2358
2300

Name	Modified	ModifiedBy	Created
Korean...	Nov 16, 2000	GREENRO...	Nov 18, 2000
	...00	GREENRO...	Nov 17, 2000
	...00	GREENRO...	Nov 16, 2000
	...00	GREENRO...	Nov 17, 2000
	...00	GREENRO...	Nov 17, 2000
	...00	GREENRO...	Nov 17, 2000
	...00	GREENRO...	Nov 17, 2000
	...00	GREENRO...	Nov 17, 2000
	...00	GREENRO...	Nov 17, 2000
French	Nov 17, 2000	GREENRO...	Nov 17, 2000
Germa...	Nov 20, 2000	GREENRO...	Nov 17, 2000
Comm...	Nov 17, 2000	GREENRO...	Nov 17, 2000
Korean...	Nov 17, 2000	GREENRO...	Nov 17, 2000
Korean...	Nov 20, 2000	GREENRO...	Nov 20, 2000
TestPa...	Nov 20, 2000	GREENRO...	Nov 20, 2000

Figure 23A


nGia 1.0 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Links Customize Links Free Hotmail Windows Media Windows
Back Forward Stop Refresh Home Search Favorites History Mail Size Print Edit Discuss Messenger
Go
Client Name
//nGia™
Help | Logout
Folders
/
Docroot
nGia
administration
Namespaces
groups
MountPoints
users
internal
new_dir
modules
handlers
embedded
templating
templates
www.test.com
admin
www.dmind.com
ngia
Applet started
Internet

Address http://demo.ngia.com:8001/Page/nGia/internal/nGia1.0

Logged in as: GREENROCKET

Name	Mo
administration	Feb 22, 2001 9:1
internal	Feb 22, 2001 9:1
modules	Feb 22, 2001 9:1
templating	Feb 22, 2001 9:1

nGia SIte Deployment

Site Root Dirctory	Deployment Start Point	Deployment Directory

Deployment was successful	Total Pages Deployed: 0

Figure 24

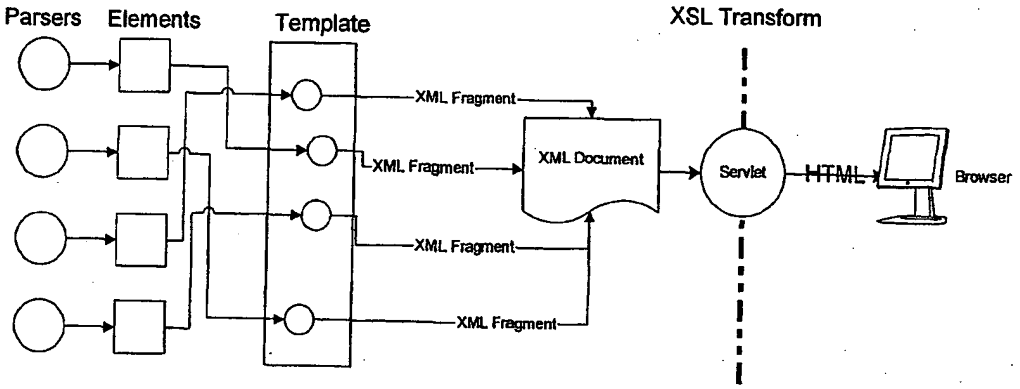

Parsers
Elements
Template
XSL Transform
XML Fragment
XML Fragment
XML Fragment
XML Fragment
XML Document
Servlet
~~HTML~~
Browser

Figure 25